Résumé hybride de flux de données par échantillonnage et classification automatique

Nesrine Gabsi*,**, Fabrice Clérot **
Georges Hébrail*

*Institut TELECOM ; TELECOM ParisTech ; CNRS LTCI
46, rue Barrault 75013 Paris
PrénomAuteur.NomAuteur@telecom-paristech.fr,
** France Telecom RD
2, avenue P.Marzin 22307 Lannion
PrénomAuteur.NomAuteur@orange-ftgroup.com

Résumé. Face à la grande volumétrie des données générées par les systèmes informatiques, l'hypothèse de les stocker en totalité avant leur interrogation n'est plus possible. Une solution consiste à conserver un résumé de l'historique du flux pour répondre à des requêtes et pour effectuer de la fouille de données. Plusieurs techniques de résumé de flux de données ont été développées, telles que l'échantillonnage, le clustering, etc. Selon le champ de requête, ces résumés peuvent être classés en deux catégories: résumés spécialisés et résumés généralistes. Dans ce papier, nous nous intéressons aux résumés généralistes. Notre objectif est de créer un résumé de bonne qualité, sur toute la période temporelle, qui nous permet de traiter une large panoplie de requêtes. Nous utilisons deux algorithmes : CluStream et StreamSamp. L'idée consiste à les combiner afin de tirer profit des avantages de chaque algorithme. Pour tester cette approche, nous utilisons un Benchmark de données réelles "KDD_99". Les résultats obtenus sont comparés à ceux obtenus séparément par les deux algorithmes.

1 Introduction

Il existe actuellement plusieurs applications qui génèrent des informations en très grande quantité. Ces applications sont issues de domaines variés tels que la gestion du trafic dans un réseau IP. Lorsque le volume de données augmente, il devient très coûteux de stocker toutes les données avant de les analyser : il est judicieux d'adopter un traitement à la volée pour ces informations. Un nouveau mode de traitement de l'information émerge. Il s'agit du traitement de flux de données. Dans (Golab et Özsu, 2003), les auteurs définissent un flux de données comme étant une séquence d'items continue, ordonnée, arrivant en temps réel avec des débits importants.

Plusieurs travaux ((Babcock et al., 2002), (Golab et Özsu, 2003), (Ma et al., 2007), (Towne et al., 2007)) montrent que les Systèmes de Gestion de Base de Données (SGBD) sont inadaptés pour ce type d'applications. Ceci est essentiellement dû à la nature continue du flux

ainsi que la volumétrie des données qui transitent. En réponse aux besoins spécifiques des applications qui traitent des flux de données, plusieurs Systèmes de Gestion de Flux de Données (SGFD), académiques et commerciaux, ont été développés : Stream (Arasu et al., 2004), Aurora (Abadi et al., 2003), etc.

Les SGFD permettent d'exprimer des requêtes continues qui s'évaluent au fur et à mesure sur un flux ou sur des fenêtres (sous ensembles finis du flux) (Babcock et al., 2002). Ces requêtes doivent être spécifiées avant l'arrivée du flux. De nouveaux besoins peuvent parfois apparaître après le passage de l'information. Dans ce cas, le système ne pourra pas répondre aux requêtes posées car toutes les données n'appelant aucun traitement sont définitivement perdues. Il est donc nécessaire dans certains cas de conserver un résumé du flux de données.

Il existe dans la littérature de nombreux travaux sur des structures de résumé (Gemulla et Lehner, 2008), (Cormode et Garofalakis, 2007), (Guha et Harb, 2005), (Guha et al., 2001), (Flajolet et Martin, 1985), (Bloom, 1970) qui sont des structures dédiées à une tâche particulière. La problématique de cet article concerne les résumés généralistes. Leur objectif est de fournir un résultat approché pour n'importe quelle analyse sur les données d'origine (requête ou fouille) posée à l'instant présent sur une fenêtre du passé. Ce domaine est encore récent et il n'existe à ce jour que peu de travaux (Csernel et al., 2006), (Aggarwal et al., 2003).

Dans ce papier, nous nous intéressons à la préservation à long terme des données du flux. Notre objectif est d'élaborer un résumé à la fois générique (opérationnel pour tout type d'application), généraliste, évolutif (adaptable aux variations des flux) et de bonne qualité (représentatif du flux d'origine) sur toute la période temporelle. Nous utilisons deux algorithmes StreamSamp (Csernel et al., 2006) et CluStream (Aggarwal et al., 2003) afin de tirer profit des avantages de ces deux approches et de créer un résumé robuste qui respecte les critères énoncés.

Nous proposons dans ce papier une méthode hybride qui permet de résumer des grands flux de données numériques. Il est possible de lancer, sur le résumé conçu, des requêtes et d'obtenir des réponses proches de celles qu'on aurait obtenues en présence de la totalité des données.

2 Résumés de flux de données

De nombreuses structures de résumé ont été développées. Certaines structures sont à vocation généraliste, alors que d'autres sont destinées à un type particulier de traitement. Nous nous intéressons dans ce papier aux résumés généralistes.

2.1 Résumé généraliste

D'après (Csernel, 2008) un résumé généraliste doit remplir quatre conditions : répondre aux requêtes posées sur n'importe quel horizon temporel fixé, traiter un large champ de requêtes (sélection, médiane, etc.), permettre des analyses supervisées des données (arbres de décision, etc.) et autoriser des tâches d'analyse exploratoire (classification, etc.).

Une autre façon de définir un résumé généraliste consiste à réaliser une organisation temporelle d'une série d'informations extraites d'un flux de données. Ces extractions peuvent être réalisées par le biais de solutions mémoires telles que l'échantillonnage, les histogrammes, etc.

2.1.1 Solutions Mémoires

Les solutions mémoires permettent de conserver des résumés soit de la totalité des flux de données, soit des résumés qui portent sur les dernières observations. Il existe deux grandes opérations d'extraction de l'information : l'échantillonnage et les histogrammes.

L'échantillonnage. L'échantillonnage est une technique directement issue des statistiques ayant pour objectif de fournir des informations sur une large population à partir d'un échantillon représentatif issu de celle-ci. Les techniques d'échantillonnages classiques ne sont pas toutes adaptées aux flux de données. En effet, elles nécessitent d'avoir l'intégralité de la base afin de sélectionner un échantillon représentatif. Dans le cadre des flux de données, cette contrainte ne peut être respectée.

De nouveaux algorithmes d'échantillonnage ont été développés. Vitter (Vitter, 1985) a introduit une technique d'échantillonnage aléatoire incrémentale. Cette technique permet de concevoir un résumé sur la totalité du flux de données. Il existe des techniques plus sophistiquées qui permettent de maintenir un échantillon aléatoire sur une fenêtre glissante (Babcock et al., 2002).

L'objectif de notre approche est de permettre des interrogations sur le passé lointain. Adapter une technique d'historisation selon le principe de Babcock serait à l'encontre de cet objectif.

Les histogrammes. La technique des histogrammes est très proche de la technique de l'agrégat. Un histogramme permet de grouper les données d'une variable du flux selon les valeurs possibles. Pour chaque groupe, une surface est construite dont la base correspond aux valeurs de ce groupe, et dont la taille est proportionnelle au nombre d'observations dans le groupe.

2.1.2 Solutions temporelles

Les solutions temporelles permettent de gérer au cours du temps les flux d'informations produits par les techniques mémoires. Elle permettent d'archiver des résumés qui couvrent l'intégralité du flux avec un espace borné.

Les fenêtres inclinées. Cette structure permet de conserver des résumés couvrant des périodes temporelles de tailles variables. Comme le montre la Figure 1, les résumés ont une taille constante mais s'étalent sur des périodes temporelles de durées variables, plus courtes pour le présent et plus longues pour le passé lointain.

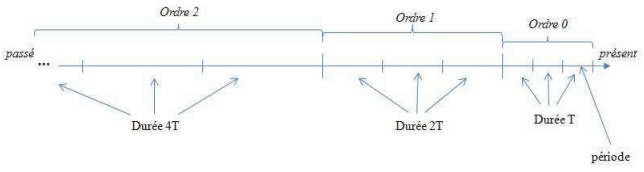

FIG. 1 – *Structure des fenêtres inclinées.*

Les systèmes de clichés. Tout comme les fenêtres inclinées, cette technique permet un traitement efficace de la dimension temporelle. L'objectif est simple, il s'appuie sur la sauvegarde à des instants réguliers de l'état du système au moment où un cliché est pris. Le cliché ainsi obtenu fournit des informations sur le traitement du flux depuis son lancement jusqu'à l'instant de sa capture. Ces clichés sont ensuite conservés selon une structure pyramidale. Comme le montre la Figure 2, cette structure privilégie les instants récents par rapport aux instants les plus anciens. Proportionnellement au temps, le nombre de clichés conservés est de plus en plus faible.

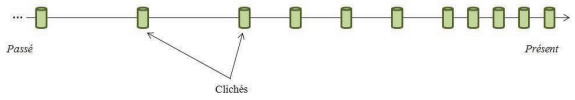

FIG. 2 – *Représentation des clichés sur la structure pyramidale.*

2.2 Clustream : approche par clustering

CluStream (Aggarwal et al., 2003) est un algorithme robuste basé sur la classification automatique des données numériques. L'idée consiste à construire un résumé du flux de données sous la forme d'un ensemble de micro-classes évolutives dont des clichés sont mémorisés régulièrement. Pour développer cet algorithme, Aggarwal s'est inspiré de l'algorithme BIRCH (Zhang et al., 1996) en utilisant la structure des CFVs (Cluster Feature Vector). Cette structure conserve des données statistiques qui résument l'ensemble des éléments d'une micro-classe (effectif total, somme des éléments de chaque variable et somme de leurs carrés). Dans (Aggarwal et al., 2003), Aggarwal ajoute aux CFVs l'extension temporelle des variables. Lorsqu'un nouvel élément du flux arrive, on calcule sa distance par rapport au barycentre de chaque micro-classe et on l'ajoute à la classe la plus proche. Lors de l'ajout d'un élément, le CFV de la micro-classe est automatiquement mis à jour sans que l'appartenance de l'élément à la classe soit mémorisée.

Cet algorithme permet de mémoriser, dans un cliché, l'état de toutes les micro-classes. Ces clichés sont ensuite conservés selon une structure pyramidale. Grâce aux propriétés mathématiques des CFVs, on peut suivre l'évolution de ces micro-classes. On effectue des opérations de soustractions sur les clichés afin d'obtenir le contenu du flux entre deux prises de clichés. Cette technique permet de fournir, dans la limite du nombre de clichés pris, une version du résumé pour n'importe quelle section du flux située entre la prise de deux clichés.

En se basant sur une technique de clustering couplée avec une structure de fenêtre pyramidale, CluStream permet de conserver des clichés représentatifs même pour des données anciennes. Ceci permet de suivre l'évolution du flux de données au cours du temps. Cependant, son inconvénient est le traitement relativement lourd des calculs de distance lorsque le débit du flux est élevé.

2.3 StreamSamp

StreamSamp est également un algorithme de résumé généraliste basé sur l'échantillonnage aléatoire du flux de données. Dès leur arrivée, les données du flux sont échantillonnées à un

taux α et placées dans des échantillons de taille T. Lorsque T est atteint, StreamSamp mémorise l'échantillon ainsi que sa date de début et fin de constitution. L'ordre 0 est associé à cet échantillon. Plus on avance dans le temps, plus le nombre d'échantillons constitués, d'ordre 0, augmente. Il est impossible de mémoriser de façon permanente tous ces échantillons.

Pour réduire l'espace, l'algorithme se base sur la structure des fenêtres inclinées. Lorsque L (nombre maximal d'échantillons par ordre) est atteint, on fusionne les deux échantillons d'ordre i les plus anciens. Un nouvel échantillon de taille T qui couvre une période temporelle, deux fois plus grande, et qui sera affecté à l'ordre $i+1$ est construit par ré-échantillonnage aléatoire. Par ailleurs, StreamSamp permet l'exploitation et l'analyse du résumé créé. Pour traiter une période temporelle, les échantillons qui appartiennent à cette période sont concaténés et pondérés de façon à conserver la même représentativité pour chaque élément de l'échantillon final.

Le résumé conçu à partir de StreamSamp a la particularité d'être petit et rapide à élaborer. Il ne dépend pas de la vitesse d'arrivée des éléments. Cependant la qualité du résumé produit pour des anciennes périodes temporelles se dégrade. En effet, les anciens éléments ont un poids croissant pour une taille d'échantillon constamment fixe. Par conséquent, si un échantillon contient des éléments récents (poids beaucoup plus faible) et des éléments anciens, ces derniers vont considérablement augmenter les erreurs dans le résultat des requêtes. L'utilisation des fenêtres inclinées ne permet donc pas de privilégier (en terme de précision) une autre période que le passé récent.

3 Approche hybride de résumé généraliste

Dans cet article nous proposons une nouvelle approche pour l'amélioration des résumés généralistes. Cette approche, intitulée "approche hybride", combine les avantages de StreamSamp et CluStream, tout en évitant leurs inconvénients. Étant donné qu'on fait intervenir CluStream, seul le cas des données numériques peut être considéré.

Comme le montre la Figure 3, le flux est d'abord envoyé dans le processus StreamSamp qui permet de conserver des échantillons aléatoires. Lorsque les échantillons ne sont plus représentatifs au sens des critères détaillés plus bas (section 3.1), ils sont envoyés dans le processus CluStream. Pour respecter la chronologie de CluStream il faut, avant l'envoi de ces échantillons, envoyer les échantillons des ordres supérieurs. L'insertion des échantillons se fait élément par élément tout en respectant leur pondération.

Le passage d'un processus vers l'autre n'est pas aléatoire mais doit respecter un certain nombre de critères. Les critères de passage définis ci-dessous permettent de déterminer quand un échantillon est représentatif ou non.

3.1 Critères de passage

Le choix des critères de passage est basé sur les propriétés intrinsèques des deux techniques : StreamSamp est guidé par le ré-échantillonnage aléatoire et CluStream par la mise-à-jour de micro-classes évolutives. Maintenir une bonne qualité pour ces deux processus permet de conserver un résumé précis et de qualité. Le premier critère (test sur la variance) surveille le ré-échantillonnage aléatoire qui entraine une dégradation du résumé de StreamSamp. Le second critère se base sur la conservation des centroides, mode de représentation de CluStream.

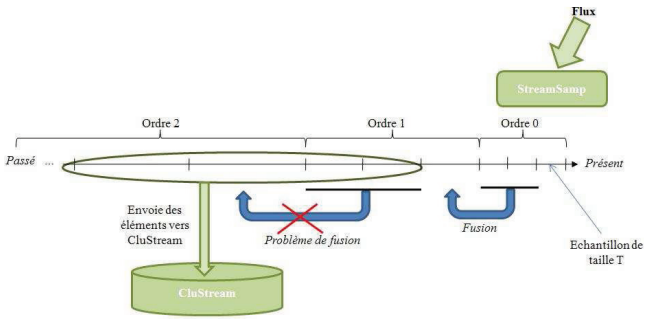

FIG. 3 – *Approche de passage de StreamSamp vers CluStream.*

3.1.1 Critère de variance

Notre objectif est de mesurer la qualité de la représentation par un échantillon sur chaque variable, ce qui revient à étudier une contrainte sur la variance. En effet, une fusion de deux échantillons d'une période donnée donne lieu à deux fois moins d'individus qui s'étalent sur une période de même durée. Le mode de représentation utilisé dans StreamSamp est ainsi de moins en moins précis, donc de moindre qualité.

Un échantillon est de bonne qualité s'il permet de répondre de façon suffisamment précise aux requêtes posées. Il doit pouvoir bien estimer les agrégats (nombre d'individus, moyenne, somme, etc.).

Puisqu'il s'agit d'un échantillonnage aléatoire, étudier la précision sur l'estimation d'un agrégat revient à calculer sa variance. Ce qui permet de quantifier l'erreur relative d'un estimateur. L'utilisation d'un critère de variance permet de fixer une borne sur l'erreur relative (*err*) ce qui revient à fixer une borne sur la variance, calculée sur l'échantillon. Dans la théorie des sondages, l'erreur relative est calculée comme suit :

$$err = \frac{\left|\widehat{F} - F\right|}{\widehat{F}} \text{ avec F : valeur de l'agrégat et } \widehat{F} \text{ : estimateur de } F$$

La question posée concerne la précision d'un échantillon E_3 issu de la fusion de deux échantillons E_1 et E_2. Pour cela, nous cherchons à estimer l'erreur sur E_3. Dans ce qui suit, on prendra l'exemple de la moyenne qui est estimée à partir de l'échantillon $(E_1 \cup E_2)$.

Vu que notre sondage est aléatoire simple, nous obtenons ainsi avec une probabilité de 95% :

$$\left|\overline{x} - \widehat{\overline{x}}(E_3)\right| \leq 2\sqrt{Var(\widehat{\overline{x}}(E_1 \cup E_2))}$$

Avec : E_1 et E_2 : les deux échantillons à fusionner, E_3 : l'échantillon résultant de la fusion, $\widehat{\overline{x}}$: l'estimateur de la moyenne calculé sur E_3 et $\overline{x}$: la vraie moyenne.

Si on vérifie l'inégalité suivante : $\frac{2\sqrt{Var(\widehat{\overline{x}}(E_1 \cup E_2))}}{\widehat{\overline{x}}(E_3)} \leq err$ alors on estime qu'on a conservé un échantillon de bonne qualité (erreur relative inférieure ou égale à *err*).

La variance de l'estimateur sur l'échantillon, est calculée comme suit :

$$Var(\widehat{\overline{x}}(E_1 \cup E_2)) = (1 - \frac{2n}{N})(\frac{1}{2n})[\frac{1}{2n-1} \sum_{k \in E_1 \cup E_2} (x_k - \overline{x})^2]$$

où n : taille d'un échantillon, N : taille de la population concernée

3.1.2 Critère de conservation des centroïdes

Dans notre approche de passage, le cycle de vie d'un échantillon est composé de deux phases complémentaires. La première étape est l'évolution de l'échantillon dans StreamSamp. La deuxième est l'intégration de cet échantillon dans le processus de CluStream.

Il est de ce fait important de prendre en considération les contraintes relatives à chaque processus. Nous avons étudié dans la section précédente la contrainte sur le fondement de StreamSamp (contrainte sur la variance). En suivant la même logique, nous étudions la conservation des centroïdes lors de l'insertion des échantillons dans CluStream. En effet, CluStream permet de regrouper les éléments "proches" dans une même micro-classe. Le processus de ré-échantillonnage aléatoire entraîne une dégradation du résumé. Conséquence, la précision sur la position des centroides se dégrade. Nous avons alors ajouté un second critère qui se base sur la conservation des centroides pour respecter CluStream.

Afin de conserver cette précision, nous calculons lors de chaque opération de ré-échantillonnage, la distance entre le centroïde (G) (calculé à partir des échantillons à fusionner ($E_1 \cup E_2$)) et le centroïde ($\overline{G}$) (calculé à partir de l'échantillon estimé (E_3)).

La distance entre (G) et ($\overline{G}$) doit être inférieure à un seuil fixé (D). Si cette condition n'est plus respectée, le processus d'échantillonnage est arrêté pour passer à CluStream.

Comme le montre la Figure 4, nous calculons à partir des individus de E_1 et E_2 le centroïde (G). Nous estimons par la suite E_3 à partir de E_1 et E_2 et nous calculons ($\overline{G}$) à partir de l'échantillon E_3.

Nous vérifions par la suite si $d^2(G, \overline{G}) \leq D$. Avec $D = \epsilon \times \sum_{E_1 \cup E_2} (d^2(\overline{x}, x_i))$, l'inertie intra-classe de l'échantillon constitué à partir de ($E_1 \cup E_2$).

4 Expérimentations

Toutes les expérimentations ont été lancées sur un ordinateur équipé d'un processeur Intel Core 2 Duo 1.66 Ghz avec 1000 Mo de mémoire. Le système d'exploitation est Windows XP Professionnel. Tous les algorithmes ont été codés dans le langage de programmation Java. Pour évaluer l'efficacité de notre algorithme, nous l'avons comparé aux algorithmes CluStream et StreamSamp. Les expériences ont été lancées sur un jeu de données réelles, KDD-CUP'99 Network Intrusion Detection stream data set. Il s'agit du même jeu de données préalablement utilisé pour l'évaluation de CluStream et StreamSamp. Ce jeu correspond à l'ensemble des connexions TCP enregistrées sur une période de deux semaines de trafic sur le réseau LAN géré par le MIT Lincoln Labs. Le jeu sur lequel nous avons mené nos expériences contient 500000 enregistrements avec 34 variables numériques. Chaque enregistrement correspond à une connexion. Une connexion fait appel à un service (http, Telnet, etc.). Les variables utilisées dans nos expérimentations sont :

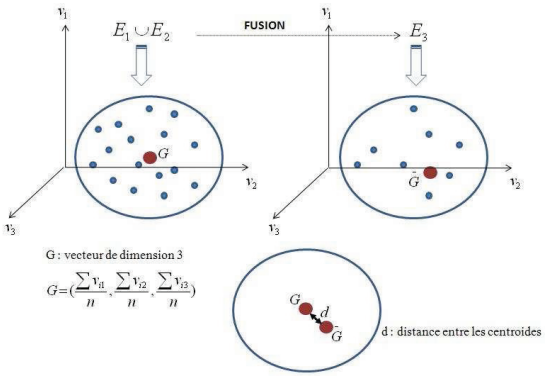

FIG. 4 – *Calcul de distance entre les centroides G et $\overline{G}$.*

- Variable 16 (Count) : Nombre de connexions émises durant les 2 dernières secondes vers la machine destinataire de la connexion courante ;
- Variable 24 (srv_diff_host_rate) : Pourcentage des connexions faisant appel au même service que la connexion courante mais vers des destinataires différents ;
- Variable 26 (dst_host_srv_count) : Nombre de connexions ayant le même service et la même destination que la connexion courante ;
- Variable 28 (dst_host_diff_srv_rate) : Pourcentage des connexions vers le même destinataire et faisant appel à des services différents que ceux de la connexion courante ;
- Variable 29 (dst_host_same_src_port_rate) : Pourcentage des connexions vers la même destination que la connexion courante et utilisant le même port.

Étude de la variabilité du flux. Pour étudier la variabilité du flux, nous avons calculé la moyenne et la variance pour chaque variable sur une fenêtre glissante de taille 5000 et avec un pas de 1000 éléments. Les résultats obtenus ont permis de vérifier que le flux de données utilisé n'est pas stationnaire pour les moments d'ordre 1 et 2. De ce fait, ce jeu de données est intéressant pour évaluer notre approche.

Dégradation des données dans StreamSamp. Cette expérience consiste à étudier la dégradation des données dans StreamSamp pour des périodes temporelles anciennes. Nous avons étudié l'évolution de l'estimation de la moyenne sur la période temporelle [0-10000]. Nous avons lancé StreamSamp ($\alpha = 30\%$, $L = 4$, $T = 50$) 500 fois pour chacune des variables étudiées. Ces paramètres ont été choisis de façon à garantir plusieurs opérations de fusions dans StreamSamp. Nous avons calculé à différents instants d'observation : (1) l'erreur relative moyenne sur les différents tirages où il existe une estimation ainsi que, (2) le pourcentage des tirages dans lesquels des estimations peuvent être effectuées. La Figure 5 montre les résultats obtenus. Nous constatons que pour chacune des variables étudiées l'erreur relative moyenne

StreamSamp	CluStream	Approche Hybride
$T = 50$	Nb- classes = 50	$err = 4.10^{-2}$
$\alpha = 30\%$	k-means = 10	$\epsilon = 10^{-4}$
$L = 4$	Instant de prise de cliché : T = 1	
	Nombre max de clichés par ordre : L = 50	

TAB. 1 – *Paramètres des expérimentations.*

augmente avec le nombre d'éléments traités. Ce qui représente une dégradation des performances de StreamSamp au cours du temps.

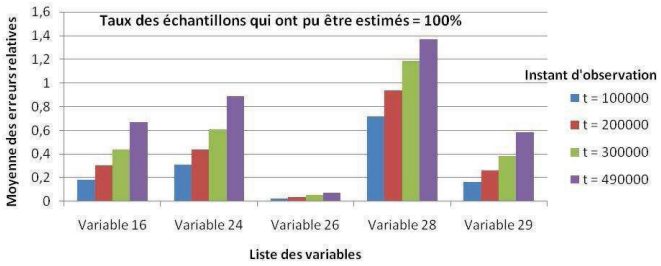

FIG. 5 – *Erreur relative moyenne sur la période [0-10000] à différents instants d'observation.*

Performance de l'approche hybride. L'objectif de cette expérience est de comparer les performances de notre approche par rapport à StreamSamp et CluStream. Pour cela, nous avons traité tout le flux de données et évalué l'erreur relative moyenne de l'estimation sur la période temporelle [0-10000]. Nous avons répété ce test 50 fois. Le Tableau 1 présente les paramètres avec lesquels nos expérimentation ont été lancées. La Figure 6 montre que pour une période temporelle assez ancienne [0-10000], évaluée à la fin du flux, l'approche hybride estime beaucoup mieux la moyenne que StreamSamp et ceci pour toutes les variables étudiées. Par ailleurs, les performances en termes de qualité d'estimation de notre approche sont proches de celles calculées par CluStream qui permet de conserver la moyenne exacte des variables.

Vitesse d'exécution. La Figure 7(a) montre les performances en termes de temps d'exécution (échelle logarithmique) accomplies pour les trois algorithmes. Il est clair que StreamSamp fournit de meilleurs résultats comparé à CluStream et à l'approche hybride. Notre approche hybride est un peu plus lente que StreamSamp mais reste beaucoup plus rapide que CluStream.

Ordre de bascule. Comme le montre la Figure 7(b), sur les 500 000 enregistrements, l'ordre de bascule varie de 1 à 4. La majorité de ces enregistrements basculent vers CluStream à l'ordre 1. Ceci va impliquer un passage "forcé" des échantillons d'ordre supérieur (plus anciens). En effet, afin de respecter la chronologie de CluStream, si un problème de fusion est détecté

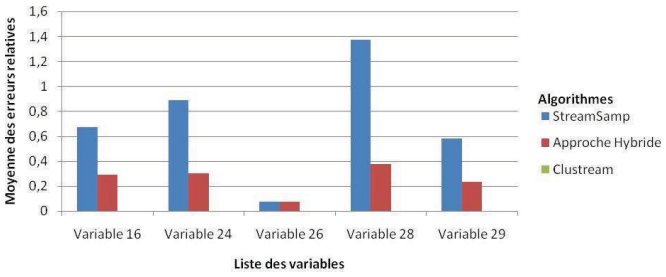

FIG. 6 – *Erreur relative moyenne sur la période [0-10000] observée en t = 500000.*

entre deux échantillons d'ordre i, alors il faut basculer les échantillons les plus anciens vers CluStream.

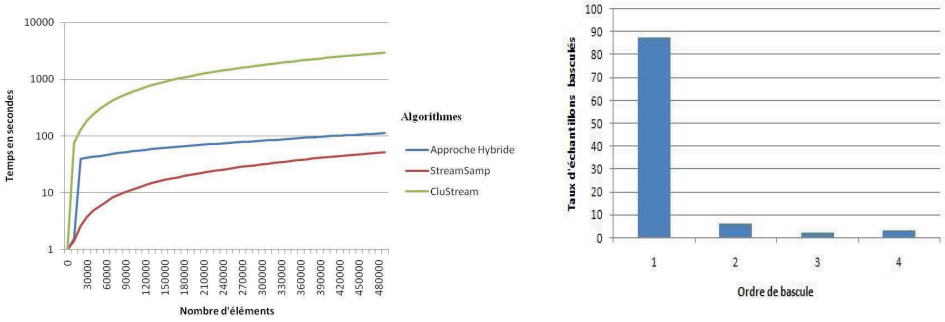

FIG. 7 – *(a)Vitesse d'exécution, (b)Taux des échantillons basculés, pour chaque ordre, vers CluStream.*

5 Conclusion et perspectives

L'objectif de cet article est de proposer une nouvelle approche pour la conception d'un résumé de flux de données. Il s'agit de développer un résumé à la fois généraliste et de bonne qualité pour toute la période temporelle.

Nous avons présenté une stratégie de passage de l'algorithme StreamSamp vers l'algorithme CluStream en nous basant sur les avantages de chacune de ces méthodes. L'instant de passage n'est pas aléatoire, il est basé sur les propriétés intrinsèques des deux techniques. Si l'un de ces critères n'est plus vérifié, le processus de fusion des échantillons de StreamSamp est arrêté et les échantillons sont envoyés vers CluStream. Cependant, avant d'envoyer ces échantillons, il faut tout d'abord faire basculer tous les échantillons des ordres supérieurs pour respecter la chronologie de CluStream. L'inconvénient avec cette approche est que nous condamnons des échantillons à passer tôt vers ce processus alors qu'ils vérifient encore les critères de passage. Des travaux sont en cours afin de développer des techniques permettant d'éviter cet inconvénient.

Une extension naturelle de ce travail concerne l'intégration des données qualitatives. Avec StreamSamp, le problème d'utilisation de variables catégorielles ne se pose pas. Concernant CluStream, il existe des extensions qui traitent les variables catégorielles (HClustream (Yang et Zhou, 2006), SCLOPE (Kok leong Ong et al., 2004)). Intégrer les données qualitatives dans notre approche est possible à condition de redéfinir les paramètres de tests pour la transition.

Références

Abadi, D. J., D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, et S. Zdonik (2003). Aurora : a new model and architecture for data stream management. *The VLDB Journal 12*(2), 120–139.

Aggarwal, C. C., J. Han, J. Wang, et P. S. Yu (2003). A framework for clustering evolving data streams. In *VLDB '2003 : Proceedings of the 29th international conference on Very large data bases*, pp. 81–92. VLDB Endowment.

Arasu, A., B. Babcock, S. Babu, J. Cieslewicz, K. Ito, R. Motwani, U. Srivastava, et J. Widom (2004). Stream : The stanford data stream management system. In *In Data-Stream Management : Processing High-Speed Data Streams*. Springer-Verlag.

Babcock, B., S. Babu, M. Datar, R. Motwani, et J. Widom (2002). Models and issues in data stream systems. In *PODS '02 : Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, New York, NY, USA, pp. 1–16. ACM.

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Commun. ACM 13*(7), 422–426.

Cormode, G. et M. Garofalakis (2007). Sketching probabilistic data streams. In *SIGMOD '07 : Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, New York, NY, USA, pp. 281–292. ACM.

Csernel, B. (2008). *Résumé Généraliste de flux de données*. Ph. D. thesis, Ecole Nationale Supérieure des Télécommunications.

Csernel, B., F. Clérot, et G. Hébrail (2006). Datastream clustering over tilted windows through sampling. *Knowledge Discovery from Data Streams Workshop (ECML/PKDD)*.

Flajolet, P. et G. N. Martin (1985). Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci. 31*(2), 182–209.

Gemulla, R. et W. Lehner (2008). Sampling time-based sliding windows in bounded space. In *SIGMOD '08 : Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, New York, NY, USA, pp. 379–392. ACM.

Golab, L. et M. T. Özsu (2003). Issues in data stream management. *SIGMOD Rec. 32*(2), 5–14.

Guha, S. et B. Harb (2005). Wavelet synopsis for data streams : minimizing non-euclidean error. In *KDD '05 : Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, New York, NY, USA, pp. 88–97. ACM.

Guha, S., N. Koudas, et K. Shim (2001). Data-streams and histograms. In *STOC '01 : Proceedings of the thirty-third annual ACM symposium on Theory of computing*, New York, NY,

USA, pp. 471–475. ACM.

Kok leong Ong, W. L., W. keong Ng, et E. peng Lim (2004). Sclope : An algorithm for clustering data streams of categorical attributes. Technical report.

Ma, L., W. Nutt, et H. Taylor (2007). Condensative stream query language for data streams. In *ADC '07 : Proceedings of the eighteenth conference on Australasian database*, Darlinghurst, Australia, Australia, pp. 113–122. Australian Computer Society, Inc.

Towne, K., Q. Zhu, C. Zuzarte, et W.-C. Hou (2007). Window query processing for joining data streams with relations. In *CASCON '07 : Proceedings of the 2007 conference of the center for advanced studies on Collaborative research*, New York, NY, USA, pp. 188–202. ACM.

Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Trans. Math. Softw. 11*(1), 37–57.

Yang, C. et J. Zhou (2006). Hclustream : A novel approach for clustering evolving heterogeneous data stream. In *ICDMW '06 : Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops*, Washington, DC, USA, pp. 682–688. IEEE Computer Society.

Zhang, T., R. Ramakrishnan, et M. Livny (1996). Birch : An efficient data clustering method for very large databases. In H. V. Jagadish et I. S. Mumick (Eds.), *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pp. 103–114. ACM Press.

Summary

Given the large volume of data generated by computer systems, storing all data before querying them is not possible. One solution is to keep a summary of the history of data streams. This history can be used to answer queries and perform data mining. Many data summarizing techniques have been already developed such as sampling, clustering, etc. According to the scope of applications, these summaries are classified into two categories: specialized summaries and generalist summaries. This paper focuses on generalist summaries. The objective is to create a good quality summary which covers a long time period and allows to process a wide range of queries. The reported work focuses on two algorithms: CluStream and StreamSamp. In order to take advantages of the benefits of each algorithm, we suggest their integration into one process. To test this approach, we use the "KDD_99" real Benchmark data. The results are separately compared to those obtained by StreamSamp and Clustream.