EGC 2007

Atelier Flux de Données

23 janvier 2007, Namur, Belgique

Organisateurs

Talel Abdessalem, ENST Paris (Talel.Abdessalem@enst.fr)
Georges Hébrail, EDF R&D et ENST Paris (Georges.Hebrail@enst.fr)

EGC 2007 - Atelier Flux de Données

Objectifs et thématiques

La taille des bases de données augmente continuellement et les sources de données, capables de produire de façon continue des flux importants de données, sont de plus en plus répandues (logs web, tickets d'appels téléphoniques, trafic dans les réseaux informatiques, capteurs physiques, etc.). Dans ce contexte, l'accès à l'intégralité des données pour des besoins de traitement ou pour leur appliquer des algorithmes de fouille de données devient problématique voire impossible. L'accès aux données est strictement séquentiel et la quantité de mémoire disponible est très inférieure au volume de données issues du flux. Ceci a conduit récemment de nombreux chercheurs à repenser les techniques de traitement de données et d'extraction de connaissances et de proposer de nouvelles approches spécifiques aux flux de données.

Cet atelier vise à rassembler les chercheurs travaillant sur les problématiques de traitement de flux de données et d'extraction de connaissances à partir de flux de données. Il leur offrira une opportunité de diffuser leurs résultats, et un espace de discussion autour des résultats et travaux actuels, des problèmes ouverts et des applications émergentes dans ce domaine.

Cet atelier couvre principalement les thématiques suivantes :

- flux de données structurées, XML, spatio-temporelles, distribuées
- requêtes continues, évaluation et optimisation
- modèles d'approximation, d'échantillonnage et de résumé
- méthodes de fouille de flux de données (classification automatique, arbres de décision, règles d'association, extraction de motifs temporels, etc.)
- détection de changement dans les flux de données évolutives
- visualisation de flux de données
- applications des approches de gestion et fouille de flux de données

Comité scientifique

- Talel Abdessalem (ENST Paris)
- Michel Adiba (IMAG)
- Fabrice Clérot (France Télécom R&D)
- Grégory Cobéna (Sophis SA)
- Alain Dessertaine (EDF R&D)
- João Gama (Université de Porto, Portugal)
- Georges Hébrail (EDF R&D et ENST Paris)
- Anne Laurent (LIRMM)
- Yves Lechevallier (INRIA)
- Florent Masseglia (INRIA)
- José Moreira (Univesité d'Aveiro, Portugal)
- Hubert Naacke (Université de Paris 6)
- Pascal Poncelet (Ecole des Mines d'Alès)
- René Quiniou (IRISA)
- Philippe Rigaux (Université de Paris 9)

TABLE DES MATIERES DES COMMUNICATIONS

Un algorithme multi-agents de clustering dynamique pour des données mobiles

Gaële Simon*, Dominique Fournier**, Bruno Mermet*

*GREYC CNRS UMR 6072, 6 Boulevard du Maréchal Juin 14050 CAEN cedex
{gaele.simon, bruno.mermet}@univ-lehavre.fr,
**LITIS EA 4051, 25 rue Philippe Lebon, BP 540 76058 Le Havre cedex
dominique.fournier@univ-lehavre.fr

Résumé. Dans cet article, nous présentons un algorithme multi-agents de clustering dynamique. Ce type de clustering doit permettre de gérer des données mobiles et donc être capable d'adapter en permanence les clusters construits. Tout d'abord sont présentées des applications potentielles de cette technique notamment pour aider à la détection d'organisations d'agents dans les SMA. Puis les contraintes spécifiques à ce type de clustering sont étudiées. Ensuite une architecture multi-agents satisfaisant ces contraintes est présentée. Elle associe un algorithme fourmis et une couche d'agents cluster s'exécutant en parallèle. Enfin nous présentons les premiers résultats expérimentaux de notre travail.

1 Introduction

1.1 Clustering d'agents

Dans cet article, nous proposons une technique de clustering dynamique de données mobiles. Cette problématique est née de l'objectif initial de nos travaux visant à permettre, au cours de l'exécution d'un système multi-agents, de détecter et de suivre des groupes d'agents ayant, pendant une période donnée, une évolution similaire du point de vue des propriétés (internes aux agents) que l'on a choisi d'observer. Chaque agent observé peut être représenté par un vecteur de propriétés évoluant dans le temps correspondant à un sous-ensemble de ses propriétés internes. Ces propriétés seront choisies en fonction du type d'application associé au système multi-agents étudié. Par exemple, ces propriétés peuvent représenter le nombre de communications de l'agent, sa valeur de renforcement (pour des agents avec des capacités d'apprentissage), ou encore sa position dans un environnement situé (par exemple des fourmis [Drogoul et al. (1995)]).

Moyennant cette représentation des agents, les groupes d'agents peuvent donc être mis en évidence via des clusters évolutifs. Plus généralement, on se trouve donc face à un problème de clustering dynamique de données mobiles qui présente les particularités suivantes :

- le cardinal de l'ensemble de données à clusteriser n'est pas constant. En effet, des agents observés et donc des données mobiles peuvent apparaître ou disparaître pendant le processus de clustering ;
- des données déjà clusterisées peuvent être modifiées du fait de l'évolution des agents correspondants.

Cela peut entraîner des modifications ou des réorganisations de l'ensemble existant de clusters. Ainsi, une méthode de clustering dynamique est nécessaire afin d'adapter continuellement l'ensemble des clusters afin qu'ils reflètent le mieux possible l'état courant des données (des agents dans notre cas). Dans la suite de cet article, nous appellerons **Donnée Mobile** ou **DM** les données que nous cherchons à clusteriser.

1.2 Applications

L'application initiale et principale de ces travaux concerne donc les systèmes multi-agents (SMA). Mais ce type de technique trouve également des applications dans d'autres domaines comme la détection et le suivi de cyclones, le suivi de flux routiers et la détection d'embouteillages, le suivi de migrations animales, etc. Dans le cadre d'un projet appelé REX, en collaboration avec le CNAM et EADS, nous allons appliquer cette technique à la gestion du retour d'expérience de l'entreprise. Dans le cadre de l'application au systèmes multi-agents, nous cherchons à détecter des organisations implicites d'agents qui peuvent apparaître pendant l'exécution du système et évoluer dans le temps. On observe souvent de tels phénomènes dans des SMA dédiés à la simulation dont le but est de refléter le comportement d'entités réelles et de gérer différents types d'interactions entre elles.

1.3 Travaux connexes

À l'heure actuelle, il existe de nombreux travaux portant sur des techniques de clustering de données non complètement connues au départ.

On trouve en particulier dans cette catégorie les algorithmes de clustering incrémental tel que celui proposé dans [Charikar et al. (1997)]. Ces techniques permettent de traiter séquentiellement les données à clusteriser (des documents dans les travaux de Charikar *et al.*) par adaptations successives des clusters. Le principal inconvénient de ces travaux par rapport à notre approche est que les données déjà clusterisées ne peuvent pas évoluer au cours du temps.

Plus récemment, sont apparues des techniques de clustering de flux de données [Barbara (2002); Guha et al. (2000)]. Un flux de données est une séquence de données volumineuses telle que la trace des actions d'un utilisateur de site web ou encore une suite de données provenant de capteurs météo. Le volume important des données empêche le stockage du flux en mémoire centrale c'est pourquoi les algorithmes de clustering opérant sur ces flux doivent traiter les données progressivement. Pour ce faire, la plupart d'entre eux travaillent sur des sous-ensembles du flux formés de plusieurs données consécutives. Le premier inconvénient de ces approches est que le nombre de clusters construits reste généralement constant. De plus, comme pour le clustering incrémental, les données déjà clusterisées ne peuvent pas évoluer au cours du temps.

On trouve enfin des algorithmes de clustering de flux de données évolutifs [Aggarwal et al. (2003); Cao et al. (2006)]. La principale différence avec la catégorie précédente est que, dans ce cas, la distribution sous-jacente des données du flux peut évoluer significativement au cours du temps. Il faut donc que les algorithmes de clustering traitant ces flux soient capables de modifier de manière importante les clusters construits, voire d'en détruire, au fur et à mesure de l'arrivée de nouvelles données, ce qui rejoint une de nos préoccupations. Malheureusement, là encore, le fait que des données déjà clusterisées puissent évoluer n'est pas pris en compte.

Les travaux les plus proches de notre problématique concernent un algorithme de clustering d'objets mobiles présenté dans [Li et al. (2004)]. Une comparaison plus spécifique de nos travaux avec cet algorithme est présentée plus loin (§ 3.3).

Critères	Clustering incrémental	Clustering de flux de données	Clustering de flux de données évolutif	Objets mobiles	Clustering dynamique
Gesion de nouvelles données[1]	*oui*	*oui*	*oui*	*non*	*oui*
Nombre de clusters fixé *a priori*	*non*	*oui*	*non*	*oui*	*non*
Données mobiles	*non*	*non*	*non*	*oui*	*oui*

TAB. 1 – *Comparaison des différents types de clustering*

Les spécificités de notre problème de clustering dynamique sont résumées dans le tableau 1.

2 Notre approche

En plus des algorithmes de clustering présentés précédemment, il existe également dans la communauté agents, des algorithmes fourmis permettant de réaliser cette tâche [Deneubourg et al. (1990)]. Dans ces algorithmes, les données sont réparties dans une grille sur laquelle se déplacent des agents fourmis qui peuvent transporter des données et les rassembler dans des tas. Ces algorithmes ont des propriétés qui semblent adaptées à la prise en compte de l'évolution des données nécessaire dans notre contexte. Malheureusement, ils ont une convergence lente, ce qui est accentué dans un contexte instable.

Pour améliorer la convergence de ces approches dans un cadre statique, N. Monmarché a proposé l'algorithme AntClass [Monmarché (2000)] qui associe successivement en quatre phases un algorithme fourmis et l'algorithme k-means. Cette approche n'étant pas compatible avec l'aspect dynamique de notre problématique comme montré dans [Simon et Fournier (2004)], nous avons décidé de ne garder que le fonctionnement élémentaire et le paramétrage des fourmis utilisées dans AntClass et d'y associer une couche d'agents cluster formant ainsi une architecture multi-agents en deux couches s'exécutant en parallèle. Dans cette architecture, chaque tas créé par les fourmis est encapsulé dans un agent cluster.

2.1 La grille

La grille permet de stocker les données non clusterisées et les tas. Les données non clusterisées peuvent avoir deux origines :

- soit elles n'ont jamais été clusterisées ;
- soit elles ont été rejetées de leur cluster initial par des agents cluster ou par des fourmis. Cela peut-être dû soit à une évolution de ces données, soit à une erreur faite par les fourmis lors de la construction du cluster.

Il est important de préciser que comme pour tous les algorithmes fourmis, les propriétés des données sont indépendantes de la position des données sur la grille.

[1] capacité de l'algorithme à traiter des données non disponibles au départ

2.2 Les agents

Deux types d'agents évoluent concurremment dans le système : les fourmis et les clusters. À chaque itération, un agent est activé aléatoirement.

2.2.1 Les Fourmis

Leur comportement est identique à celui spécifié dans AntClass pendant la première étape de l'algorithme. Nous résumons rapidement ce comportement, plus de détails étant disponibles dans [Monmarché (2000); Coma et al. (2003)]. Les fourmis se déplacent sur la grille et peuvent :

1. prendre une donnée :
   (a) en emmenant une donnée isolée trouvée dans une cellule,
   (b) en ôtant la donnée la plus dissimilaire d'un tas ;
2. déposer une donnée :
   (a) sur une cellule libre,
   (b) sur une donnée isolée et ainsi créer un nouveau tas,
   (c) sur un tas existant.

La réalisation de ces actions est modulée par des probabilités qui introduisent une forme d'indéterminisme dans le comportement des fourmis, propriété indissociable des algorithmes biomimétiques.

Notre architecture reposant sur le même algorithme fourmis qu'AntClass, nous avons conservé le même processus de paramétrage à l'exception du paramètre TREMOVE utilisé pour l'action (1b). En effet, les expérimentations de notre approche réalisées sur des données statiques et présentées dans [Simon et Fournier (2004)] ont montré une trop grande sensibilité à ce paramètre ce qui nous a conduit à définir une nouvelle mesure de comparaison des données. Cette mesure, prenant en compte la dispersion des distances des données au centre de gravité du cluster permet de réduire le nombre de paramètres à fixer tout en augmentant la stabilité du clustering. Pour cela, nous supposons que la répartition des données d'un cluster construit suit une loi normale représentée par une variable R_C. Dans ce cadre, en considérant μ la moyenne et σ l'écart-type de R_C, par définition, on sait que 99,7% des données se trouve dans l'intervalle $I_3 = [\mu - 3\sigma, \mu + 3\sigma]$. Ce qui nous a conduit à redéfinir le comportement des fourmis lors du retrait d'une donnée d'un tas de la manière suivante : si la distance de la donnée la plus dissimilaire du tas à son centre n'appartient pas à l'intervalle I_3, la fourmi l'enlève.

En outre, nous avons également modifié le mécanisme de dépot d'une donnée dans un tas. En effet, les expérimentations ont montré que ce mécanisme favorisait parfois la progression de clusters erronés initialisés avec deux données très distantes. On a donc recherché un critère d'agrégation plus restrictif. Par conséquent, nos fourmis déposent désormais une donnée dans un tas si la distance de cette donnée au centre du tas se trouve dans l'intervalle $I_2 = [\mu - 2\sigma, \mu + 2\sigma]$.

2.2.2 Les Clusters

Avec AntClass, après la première phase de travail des fourmis, il reste de nombreuses données isolées et, souvent, de trop nombreux tas ont été formés. C'est pourquoi, k-means est utilisé à la suite des fourmis pour améliorer les résultats. Dans notre approche, ce problème est traité en continu par les agents cluster.

Un agent cluster encapsule un tas C (i.e. un cluster) créé par les fourmis. C est défini par un quadruplet (G_C, R_C, V_C, S_C) où G_C est son centre, R_C son rayon, V_C son volume et

S_C l'ensemble des données qu'il contient. Dans la suite, C désigne aussi bien le cluster que l'agent.

Comportement global L'algorithme général du comportement d'un agent cluster est présenté en figure 1.

```
tant_que (taille suffisante) faire
   si (agressé)
     si (intersection vide) fuite par relâchement
     sinon fuite par agrégation
   sinon
      si (il existe gêneur) agression
     sinon tentative rejet autonome
fin_tant_que
```

FIG. 1 – *Algorithme de comportement des agents Clusters*

Dès qu'une fourmi dépose la donnée qu'elle porte dans un tas ou quand elle retire une donnée d'un tas, l'agent cluster associé doit mettre à jour son quadruplet. De plus, les agents cluster peuvent également se développer en attaquant d'autres agents cluster perçus comme gênant leur propre développement (principe d'attaque/fuite reposant sur le mécanisme d'éco-résolution décrit dans [Ferber (1995)]. On considère qu'un agent cluster C_{obs} *gène* un autre agent cluster C_{att} quand le volume de C_{att} est en intersection avec celui de C_{obs}. De plus, nous supposons que C_{att} attaque C_{obs} seulement si sa taille est au moins égale à 20% de la taille de C_{obs} (pour éviter une trop grande multiplication des agressions). Si une telle intersection des volumes se produit, c'est que probablement l'état des clusters n'est plus en adéquation avec l'état courant des données. Ils doivent donc être mis à jour ce qui se fait par fusion complète ou partielle lors de la fuite de l'agent agressé (voir section suivante). Comme le montre l'algorithme, la gestion d'une fuite est toujours prioritaire. Le but principal de ce comportement d'attaque/fuite est de compenser la tendance des fourmis à construire trop de tas et de prendre en compte l'évolution des données.

Les agents cluster doivent également détecter l'évolution des données qu'ils contiennent. Si une donnée devient trop distante du centre de son tas, l'agent cluster associé la rejette dans la grille et met à jour son quadruplet (voir *fuite par relâchement* dans l'algorithme). Comme les agressions et fuites sont relativement nombreuses, le rejet autonome reste marginal. Ce mécanisme permet d'établir une certaine concurrence entre les fourmis et les clusters améliorant la convergence globale du processus de clustering. Pour évaluer si une donnée est trop éloignée du centre de son tas, un agent cluster emploie la même mesure que celle utilisée par les fourmis (§ 2.2.1).

Détails du processus d'attaque/fuite Si un cluster appelé C_{att} a un voisin gêneur C_{obs}, C_{att} attaque C_{obs} et ce dernier doit fuir. Soient $\delta(x, y)$ la distance euclidienne entre 2 données x et y et $diss(C)$ la donnée la plus dissimilaire du centre de C. Pour gérer la fuite de C_{obs}, on considère deux situations suivant que des données de C_{obs} se trouvent spatialement dans le volume $V_{C_{att}}$ ou non. Quand de telles données existent, elles sont ajoutées au tas de C_{att} (*fuite par agrégation* dans l'algorithme). Quand il n'y en a pas, C_{obs} se sépare progressivement de chacun de ses $diss(C_{obs})$ jusqu'à ce que l'intersection des volumes disparaisse (*fuite par relâchement* dans l'algorithme). Si $\delta(diss(C_{obs}), G_{C_{att}}) < R_{C_{att}} + 2\sigma$ alors $diss(C_{obs})$ est

mise dans C_{att} et sur la grille sinon. Cette règle d'agrégation des données au cluster agresseur lors d'une fuite est identique au mécanisme de dépot d'une donnée par une fourmi dans un tas (§ 2.2.1). Elle a pour but de renforcer l'homogénéité de C_{att} et d'éviter la construction de tas trop grands et difficiles à dissocier.

Naissance et mort d'un agent cluster Une fourmi peut déposer sa donnée sur une cellule contenant une donnée isolée, créant ainsi un nouveau tas donnant naissance à un agent cluster. Celui-ci survit tant que son tas contient suffisamment de données (*taille suffisante* dans l'algorithme), à moins qu'il ne meure lors d'une fuite. De plus, juste avant sa mort, un agent cluster rejette tout ou partie de ses données sur la grille. Ce mécanisme permet aux fourmis de construire de nouveaux clusters plus représentatifs des nouvelles données ou du nouvel état des données consécutif à leur évolution.

3 Prototypes et expérimentations

3.1 Présentation de la plateforme d'expérimentations

Afin de tester et d'évaluer les performances de notre algorithme de clustering dynamique, nous avons développé 2 plateformes interconnectables en JAVA en utilisant l'environnement de développement de SMA MADKIT [Gutknecht et Ferber] :

- une plateforme de simulation ;
- une plateforme de clustering dynamique ;

La plateforme de simulation permet de générer des jeux de données mobiles spécifiés par des scénarios. Une population est considérée comme étant un ensemble de DM qui vont, *a priori*, évoluer de manière similaire. Une population doit donc être normalement perçue, au niveau du clustering, comme un cluster qui va évoluer au cours du temps. Chaque DM de la population est implantée via un agent MADKIT. Les populations ainsi que leur évolution sont spécifiées à l'aide de scénarios. Un scénario contient les informations suivantes : le nombre de populations, le nombre d'individus (de données mobiles) dans chaque population, le nombre de propriétés décrivant chaque DM (de type réel, elles vont être manipulées dans la plateforme sous forme de coordonnées afin de faciliter la visualisation des résultats), une temporisation permettant de contrôler la vitesse d'évolution des populations, la date d'apparition des populations ainsi que leurs durées d'évolution, la trajectoire de chaque population

La trajectoire d'une population est considérée comme étant la spécification de son déplacement dans l'espace et dans le temps. Sa définition comporte donc les points suivants :

- définition des intervalles d'initialisation des propriétés des DM
- spécification des propriétés soumises à évolution de chaque DM
- définition du déplacement, via des points de contrôles, des propriétés soumises à évolution. Un point de contrôle est un couple (date, position) spécifiant la position moyenne d'une population à un moment donné de la simulation.

Lorsque le simulateur démarre l'exécution d'un scénario, on peut en parallèle lancer la plateforme de clustering et, ainsi, visualiser en même temps les populations réelles et les clusters détectés.

3.2 Expérimentations

3.2.1 Objectifs

Dans cette partie, nous présentons les expérimentations réalisées afin de tester et d'évaluer notre algorithme de clustering dynamique. L'objectif de ces expérimentations est notamment

d'évaluer la capacité de l'algorithme à détecter et suivre correctement des clusters par rapport à un scénario donné.

3.2.2 Scénarios utilisés

Dans ce paragraphe sont présentés quelques scénarios utilisés lors des expérimentations. Les scénarios choisis font essentiellement varier les trajectoires, parfois le nombre de clusters. Chaque scénario est résumé par une représentation graphique (FIG. 2) dans laquelle, chaque population est représentée par un cercle. Lorsque la population se trouve à l'état initial (c'est-à-dire avant le début de son évolution), le cercle est en gras. Lorsque la population se trouve dans un état intermédiaire ou final, le cercle est en pointillés. Enfin, la trajectoire de chaque population est représentée par des flèches. Dans la figure du scénario 4C, ne figure pas de cercle en pointillés car les positions initiale et finale de chaque population sont identiques ; chaque population se déplace donc le long d'un carré.

Dans chaque scénario, une population contient 100 individus répartis suivant une loi normale sauf pour le scénario 4 où les tailles des 2 populations sont de 50 et 150. On notera que, pour les premiers tests, on s'est limité à deux propriétés par DM, ce qui simplifie en particulier les visualisations pour l'analyse des résultats.

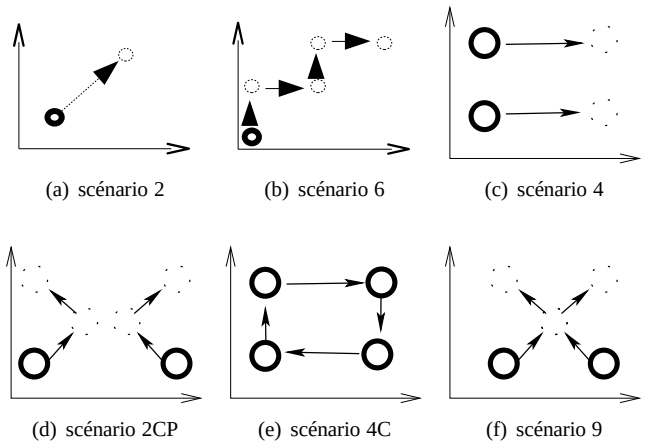

FIG. 2 – *Schéma des différents scénarios*

3.2.3 Configuration de la plateforme de clustering

L'évaluation de la plateforme de clustering a entre autres, pour but, d'étudier l'influence de différents paramètres sur la qualité des résultats produits. Au niveau de la couche des clusters, différents paramètres interviennent :

- condition d'agression pour un cluster : dans la plupart des tests, un agresseur doit au moins être aussi gros que 20% de la taille de l'agressé pour pouvoir réaliser l'agression.
- critère de survie d'un cluster : c'est le nombre minimal de données qu'un cluster doit contenir pour survivre. Dans les tests, ce nombre est de 2.
- date de démarrage des clusters par rapport aux fourmis : l'activation des agents clusters démarre, dans ces tests, 5000 itérations après les fourmis de manière à ce que ces der-

nières aient le temps de construire suffisamment de tas significatifs en taille ; à partir de cette date, fourmis et clusters travaillent simultanément.

3.2.4 Difficultés d'analyse des résultats

Un problème à étudier est celui de l'évaluation du résultat d'un clustering dynamique c'est-à-dire définir des critères sur le résultat qui permettent de répondre à la question suivante : le résultat du clustering est-il bon ou mauvais par rapport au scénario ?

On ne peut en effet seulement utiliser les critères habituels évalués sur le résultat final ; il faut prendre en compte l'aspect temporel dans l'analyse du résultat afin de pouvoir caractériser une évolution de clusters. Pour le moment, nous avons utilisé trois types de courbes temporelles : nombre de données agrégées, nombre de clusters et pureté moyenne des clusters. Pour des raisons de place, ces courbes ne seront pas présentées ici. C'est pourquoi, dans la section suivante, l'évaluation des résultats est présentée en utilisant des mesures moyennes sur le déroulement du processus de clustering. Ces mesures tiennent néanmoins compte des variations éventuelles du nombre de populations réelles au cours du scénario. Il est à noter qu'elles ne sont pas toujours suffisantes pour caractériser complètement le résultat.

3.2.5 Résultats

Dans le tableau 2 sont présentés quelques résultats d'expérimentations. Il est important de noter que dans ces statistiques, un tas de 2 données est compté comme un cluster. Pour chaque test, les mesures suivantes sont utilisées :

- #pops : nombre de populations dans le scénario
- #données : nombre total de données mobiles dans le scénario
- NmoyC : moyenne du rapport entre le nombre de clusters détectés et le nombre de populations réelles (plus le chiffre est proche de 1, meilleur est le résultat).
- NmoyDC : nombre moyen de données agrégées (c'est-à-dire appartenant à un cluster) au cours de la simulation
- TpsNbP : pourcentage de temps de la simulation où le nombre de clusters détectés est égal au nombre de populations du scénario
- PM : moyenne de la pureté moyenne des clusters.

Le scénario 11v1 reprend le scénario 2 auquel sont ajoutées 33 DM supplémentaires réparties aléatoirement dans la totalité de l'espace et correspondant à du bruit.

Test	#pops	#données	NMoyC	NMoyDC	TpsNbP	PM
2	1	100	1,1	96,6	95,3	1
6	1	100	1,0	96,1	97,2	1
4	2	200	1,2	181,5	82,1	1
2CP	2	200	1,15	180,0	84,9	0.99
9	2	200	2,5	132,6	35	0.7
4C	4	400	1,8	281,4	25,3	0.99
11v1	1+bruit	133	1,5	93,0	77,0	0.97

TAB. 2 – *Résultats des expérimentations*

Les valeurs de pureté montrent que de manière générale, les clusters créés sont bien formés c'est-à-dire qu'ils contiennent des données correspondant à une même population. Le ratio

moyen du nombre de clusters, quoique généralement trop grand, montre que le nombre de clusters détectés est proche de la réalité. En effet, l'analyse des courbes du nombre de clusters construits montre que celui-ci reste proche du nombre courant de populations moyennant une courte phase d'adaptation de l'algorithme de clustering (au cours de laquelle le nombre de clusters est important). Le nombre moyen de données agrégées montre, en revanche, que des données restent non clusterisées même si leur nombre est en général faible. La trajectoire des populations n'influence pas la qualité des résultats. De plus, les bons résultats du scénario 6 montrent que l'algorithme n'a pas été perturbé par les données bruitées.

En revanche, l'algorithme donne de mauvais résultats sur le scénario 9. Ceci est lié au fait que dans ce scénario, les 2 populations se retrouvent à mi-parcours temporairement superposées. Au niveau du clustering, cela provoque, à juste titre, une fusion des 2 clusters associés aux 2 populations à ce moment là. Le problème est que, lorsque dans la simulation, les 2 populations se re-séparent, l'unique cluster continue à exister au lieu de se dissocier.

Pour le scénario 4C, les résultats, sont moins bons qu'avec les scénarios à 1 ou 2 populations. En effet, plus les données et populations sont nombreuses, plus la probabilité pour les fourmis de créer des clusters est élevée. Enfin, l'implantation actuelle de la plateforme doit être améliorée car le mode d'activation des populations, fourmis et clusters, n'est pas suffisamment équitable, ce qui dégrade les performances lorsque le nombre de données augmente.

3.3 Comparaison avec le clustering d'objets mobiles

Le clustering d'objets mobiles [Li et al. (2004)] permet d'effectuer du clustering sur des données qui évoluent au cours du temps. L'algorithme repose sur l'utilisation de micro-clusters mobiles qui est une adaptation de la notion de micro-cluster définie dans [Zhang et al. (1996)].

Un des avantages de cet algorithme est son efficacité dans la mise à jour des micro-clusters. En effet, si les micro-clusters sont assez stables, la mise à jour peut se faire uniquement via les profils qui résument l'état du micro-cluster. De plus, la prise en compte de la vitesse comme propriété faisant partie intégrante des données à clusteriser est aussi un aspect intéressant.

Le principal inconvénient de cette approche par rapport à notre problématique est la nécessité d'utiliser conjointement un algorithme de clustering statique. Les auteurs utilisent en l'occurrence K-Means ce qui impose donc de fournir le nombre de clusters recherchés. L'autre inconvénient majeur est que l'algorithme ne permet pas de prendre en compte l'arrivée de nouvelles données en cours de route.

4 Conclusion et perspectives

Dans cet article, nous avons présenté un algorithme de clustering dynamique de données mobiles (ou d'agents) reposant sur une architecture multi-agents. Les premières expérimentations sur des données mobiles montrent de bons résultats sur le suivi de différents types d'évolutions des données.

Néanmoins de nouvelles solutions doivent être apportées, notamment pour le problème de la dissociation de clusters apparu dans l'expérimentation sur le scénario 9 (voir § 3.2.5). Une première piste va consister à introduire la vitesse des données dans le processus de clustering [Li et al. (2004)]. Par ailleurs, nous allons supprimer la grille au niveau des fourmis. Enfin, d'un point de vue implantation, nous travaillons actuellement sur une nouvelle version des plateformes permettant un meilleur contrôle sur l'activation des différents agents. A plus long terme, notre objectif est d'appliquer cette méthode à différents problèmes concernant notamment l'émergence dans les systèmes multi-agents.

algorithme multi-agents de clustering dynamique

Références

Aggarwal, C. C., J. Han, J. Wang, et P. S. Yu (2003). A framework for clustering evolving data streams. In *VLDB*.

Barbara, D. (2002). Requirements for clustering data streams. *SIGKDD Explorations 3*(2), 23–27.

Cao, F., M. Estery, W. Qian, et A. Zhou (2006). Density-based clustering over an evolving data stream with noise. In *SIAM Conference on Data Mining*, pp. 11 pages.

Charikar, M., C. Chekuri, T. Feder, et R. Motwani (1997). Incremental clustering and dynamic information retrieval. In *STOC'97 : Prodeedings of the twenty-ninth annual ACM symposium on Theory of computing*, pp. 626–635.

Coma, R., G. Simon, et M. Coletta (2003). A multi-agent architecture for agents clustering. In *Proceedings of 4th International Workshop on Agent-Based Simulation (ABS'03*. SCS Publishing House.

Deneubourg, J. L., S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, et L. Chretien (1990). The dynamics of collective sorting : robot-like ant and ant-like robots. In J.-J. Meyer et S. Wilson (Eds.), *Proceedings of the First International Conference on Simulation of Adaptative Behavior*.

Drogoul, A., B. Corbara, et S. Lalande (1995). *artificial societies : The Computer Simulation of Social Life*, Chapter MANTA : new experimental results on the emergence of (artificial) ant societies. Conte and Gilbert editors.

Ferber, J. (1995). *Les systèmes multi-agents*. InterEditions.

Guha, S., N. Mishra, R. Mortwani, et L. O'Callaghan (2000). Clustering data streams. In *IEEE Annual Symposium on Foundations of Computer Science*, pp. 359–366.

Gutknecht, O. et J. Ferber. Madkit (www.madkit.org).

Li, Y., J. Han, et J. Yang (2004). Clustering moving objects. In *KDD (Knowledge Discovery in Databases)*, pp. 617–622.

Monmarché, N. (2000). *Algorithmes de fourmis artificielles : applications à la classification et à l' optimisation*. Ph. D. thesis, Université de Tours, France.

Simon, G. et D. Fournier (2004). Agents clustering with ants. In *Proceedings of 5th International Workshop on Agent-Based Simulation (ABS'04)*, pp. 147–152. SCS Publishing House.

Zhang, T., R. Ramakrishnan, et M. Livny (1996). Birch : an efficient data clustering method for very large databases. In *SIGMOD (International Conference on Management of Data)*.

Summary

In this paper, a multi-agents algorithm for dynamic clustering is presented. This kind of clustering is intended to manage mobile data and so, to be able to continuously adapt the built clusters. A multi-agents architecture satisfying these constraints is described. It combines an ants algorithm with a cluster agents layer which is executed simultaneously. The first experimental results of our work are presented.

Limites d'une approche incrémentale pour la segmentation de séquences dans les flux

Alice Marascu, Florent Masseglia

INRIA Sophia Antipolis,
Equipe AxIS,
2004 route des Lucioles - BP 93,
06902 Sophia Antipolis, France
{Alice.Marascu,Florent.Masseglia}@sophia.inria.fr

Résumé. Les flots de données séquentielles (flux) se trouvent impliqués dans des domaines de plus en plus nombreux. Dans un processus de fouille appliqué sur un flux, l'utilisation de la mémoire est limitée, de nouveaux éléments sont générés en permanence et doivent être traités le plus rapidement possible, aucun opérateur bloquant ne peut être appliqué sur les données et celles-ci ne peuvent être observées qu'une seule fois. A l'heure actuelle, il existe très peu de méthodes de segmentation non supervisée des séquences dans un flux. Notre objectif, dans cet article, est d'étudier l'aspect incrémental que peut prendre un tel traitement. Pour cela, nous allons comparer une méthode de segmentation incrémentale des séquences d'un flux à une méthode antérieure non-incrémentale. Nos expérimentations montrent que la complexité d'une approche incrémentale n'est pas forcément payante.

1 Introduction

Depuis peu, des applications émergentes comme (entre autres) l'analyse du trafic réseaux, la détection de fraude ou d'intrusion, la fouille de clickstream[1] ou encore l'analyse des données issues de capteurs ont introduit de nouveaux types de contraintes pour les méthodes de fouille. Ces applications ont donné lieu à une forme de données connues sous le nom de "flux". Dans le contexte des flux l'utilisation de la mémoire doit être réduite, les données sont générées de manière continue et très rapide, les opérations bloquantes ne sont pas envisageables et, enfin, les nouvelles données doivent être prises en compte aussi vite que possible. Dans ce domaine, l'approximation a rapidement été reconnue comme un facteur clé pour fournir des motifs à la vitesse imposée par l'application Garofalakis et al. (2002); Teng et al. (2003); Giannella et al. (2003). Dans Marascu et Masseglia (2006b) nous avons proposé SCDS, une méthode d'extraction de motifs séquentiels dans les flux. Le flux y est découpé en batches (ou "paquets") de taille fixe traités l'un après l'autre de manière indépendante. Parmi les différentes fonctionnalités de cette méthode, la segmentation des séquences est utilisée comme préalable à l'extraction des motifs fréquents. Nous y avons montré l'efficacité d'une approche

[1] clickstream : flot de requêtes d'un utilisateur sur un site Web

centroïde pour la segmentation des séquences. Notre objectif y était d'extraire des classes de comportements (comportements qui seront représentés sous forme de séquences) à partir des flots de données d'usage d'un site Web. Dans cet article, nous proposons d'ajouter à SCDS une mémoire des batches précédents lors du traitement du flux. Pour cela, le représentant de chaque classe est conservé en mémoire. Lors du passage au batch suivant, les séquences seront classées en tenant compte de la segmentation précédente. Nous avons comparé les résultats de la segmentation dans chacun des deux cas (avec ou sans prise en compte du batch précédent) et nous présentons dans cet article nos conclusions à partir de nos expérimentations.
Cet article est organisé de la manière suivante : tout d'abord nous présentons les concepts d'extraction de motifs séquentiels dans la section 2. Ensuite la section 4 expose la technique que nous avons développée dans ce travail afin de proposer une segmentation des séquences issues d'un flux. Nous proposons des expérimentations en section 5 avant de conclure. Pour un état de l'art sur l'extraction de séquences dans les flux de données, nous encourageons le lecteur à consulter les travaux de Giannella et al. (2003); MAIDS; Raissi et al. (2005); Marascu et Masseglia (2006b).

2 Définitions

2.1 Motifs séquentiels

Ce paragraphe expose et illustre la problématique liée à l'extraction de motifs séquentiels dans de grandes bases de données. Il reprend les différentes définitions proposées dans Agrawal et Srikant (1995); Srikant et Agrawal (1996); Masseglia et al. (1998). La notion de séquence est définie de la manière suivante :

Définition 1 *Une transaction constitue, pour un client C, l'ensemble des items achetés par C à une même date. Dans une base de données client, une transaction s'écrit sous forme d'un triplet : <id-client, id-date, itemset>. Un itemset est un ensemble non vide d'items noté $(i_1 i_2 \dots i_k)$ où i_j est un item (il s'agit de la représentation d'une transaction non datée). Une séquence est une liste ordonnée, non vide, d'itemsets notée $< s_1 s_2 \dots s_n >$ où s_j est un itemset (une séquence est donc une suite de transactions avec une relation d'ordre entre les transactions). Une séquence de données est une séquence représentant les achats d'un client. Soit $T_1,\ T_2, \dots,\ T_n$ les transactions d'un client, ordonnées par date d'achat croissante et soit itemset(T_i) l'ensemble des items correspondants à T_i, alors la séquence de données de ce client est $< itemset(T_1)\ itemset(T_2) \dots\ itemset(T_n) >$.*

Exemple 1 *Soit C un client et S=$< (3)\ (4\ 5)\ (8) >$, la séquence de données représentant les achats de ce client. S peut être interprétée par "C a acheté l'item 3, puis en même temps les items 4 et 5 et enfin l'item 8".*

Définition 2 *Le $support$ de s, noté $supp(s)$, est le pourcentage de toutes les séquences dans D qui supportent (contiennent) s. Si $supp(s) \geq minsupp$, avec une valeur de support minimum $minsupp$ fixée par l'utilisateur, la séquence s est dite fréquente.*

Dans cet article nous considérons des flots de données séquentielles. Il s'agit donc à chaque instant i d'entrées de la forme $[C,\ T_1,\ T_2, \dots,\ T_n]$ qui représentent les transactions T_1 à T_n du client C à l'instant i.

La figure 1 illustre un ensemble de navigations Web à analyser. A chaque client correspond une suite de "dates" (événements) et la traduction de l'URL demandée par ce client à cette date. Dans notre contexte, nous considérons que les données sont trop volumineuses pour être stockées dans un fichier log. Nous travaillons donc sur une unité différente, le "batch", qui est une coupe faite dans le flux (une fenêtre sur le flux à un instant t).

	Date1	Date2	Date3	Date4	Date5
Client1	10	30	40	20	30
Client2	10	30	60	20	50
Client3	10	70	30	20	30

FIG. 1 – *Exemple de batch issu du flux*

L'objectif est alors de déterminer, grâce à une phase d'extraction, les séquences de ce batch, qui peuvent être considérées comme fréquentes selon la définition 2. Les résultats obtenus sont du type `< ( 10 ) ( 30 ) ( 20 ) (30 ) >` (ici avec un support minimum de 66% et en appliquant les algorithmes de fouille de données sur le fichier représenté par la figure 1). Ce dernier résultat, une fois re-traduit en termes d'URLs, confirme la découverte d'un comportement commun à $minSup$ utilisateurs et fournit l'enchaînement des pages qui constituent ce comportement fréquent.

3 SCDS : une méthode d'extraction de motifs séquentiels fréquents dans les flux

Dans Marascu et Masseglia (2006a) nous avons proposé SCDS, une méthode d'extraction de motifs séquentiels fréquents dans les flux. Dans cette section nous proposons une vue synthétique sur le principe de SCDS, qui repose sur les étapes suivantes :

1. **Découpage du flux en batches de taille fixe**. Les batches contiennent le même nombre de transactions, mais le nombre de séquences peut varier en fonction de leur taille. Le reste de la description de SCDS concerne le traitement de chaque batch.
2. **Segmentation des séquences contenues dans le batch**. Cette étape est basée sur la plus longue sous-séquence commune entre deux séquences (PLSC) de la définition 3. L'algorithme est initialisé avec une seule classe, qui contient la première navigation. Ensuite, pour chaque navigation n dans le batch, n est comparée avec chaque cluster c via son représentant (ou centroïde) ς_c. Dans SCDS, le centroïde ς_c du cluster C_i est le résultat de l'alignement des séquences de C_i. La méthode d'alignement est expliquée dans la section 4.1. Le cluster C_i qui présente le centroïde le plus compatible avec n la reçoit.
3. **Extraction des séquences fréquentes**. A la fin du traitement d'un batch, les centroïdes sont considérés comme des représentants de chaque cluster et sont utilisés pour résumer le batch.
4. **Stockage et gestion des séquences**. Un enjeu majeur de l'extraction de motifs dans les flux se situe dans la gestion de l'historique des motifs extraits. Le sujet de cet article étant principalement la méthode d'extraction nous ne détaillerons pas cet aspect ici.

Définition 3 Soient s_1 et s_2 deux motifs séquentiels. Soit $|PLSC(s_1, s_2)|$ la longueur de la plus longue sous-séquence commune entre s_1 et s_2. La distance $dist(s_1, s_2)$ entre s_1 et s_2 est définie de la manière suivante : $dist = 1 - \frac{2 \times |PLSC(s_1,s_2)|}{longueur(s_1)+longueur(s_2)}$.

Notre objectif, dans ce travail, est de montrer les limites d'une méthode de segmentation incrémentale appliquée dans le contexte de SCDS. Nous avons défini et implémenté la méthode ICDS (Incremental Clustering in Data Streams). ICDS reprend les étapes principales de SCDS mais apporte une modification à la segmentation des séquences contenues dans un batch. Nous détaillons les différences dans la section 4.

4 ICDS : principe général

Notre méthode est basée sur un environnement de découpage du flux en "batches" de taille fixe. Soient $B_1, B_2, ...B_n$, les batches et B_n, le batch courant. Le principe de ICDS est de segmenter les séquences contenues dans chaque batch b de $[B_1..B_n]$.

Dans le but d'obtenir une segmentation des navigations aussi rapidement que possible, notre approche fonctionne de la manière suivante : l'algorithme est initialisé avec une seule classe, qui contient la première navigation. Ensuite, pour chaque navigation n dans le batch, n est comparée avec chaque cluster c. Soit c le cluster dont le centroïde est le plus proche de n, alors n est insérée dans c. Si n n'est insérée dans aucun cluster, alors un nouveau cluster est créé et n est insérée dans ce nouveau cluster. Trois étapes sont donc essentielles dans ce processus. La première est le calcul du centroïde ς_c du cluster c. Ce calcul est détaillé dans la section 4.1. Ensuite pour comparer la séquence de navigation n avec le cluster c, nous proposons de définir la similitude entre n et le centroïde de c. Cette étape est expliquée dans la section 4.2. Enfin, le centroïde d'un cluster sera mémorisé du batch B_n au batch B_{n+1} et servira de base pour une segmentation semi-supervisée.

4.1 Calcul du centroïde

Le centroïde du cluster est déterminé par une technique d'alignement appliquée sur le cluster (comme Kum et al. (2003); Hay et al. (2002) l'ont déjà utilisée pour la fouille de bases de données statiques). A l'initialisation d'un cluster son centroïde est la séquence unique qu'il contient.
L'alignement des séquences renvoie une séquence alignée du type : $SA =< I_1 : n_1, I_2 : n_2, ..., I_r : n_r >: m$. Dans cette représentation, m représente le nombre total de séquences impliquées dans l'alignement. I_p ($1 \leq p \leq r$) est un itemset représenté sous la forme ($x_{i_1} : m_{i_1}, ..., x_{i_t} : m_{i_t}$), où m_{i_t} est le nombre de séquences qui contiennent l'item x_i à la p^{eme} position dans la séquence alignée. Enfin, n_p est le nombre d'occurrences de l'itemset I_p dans l'alignement. L'exemple 2 décrit le processus d'alignement de quatre séquences. À partir de deux séquences, l'alignement commence par insérer des itemsets vides (au début, au milieu ou à la fin des séquences) jusqu'à ce que les deux séquences contiennent le même nombre d'itemsets.

Exemple 2 *Considérons les séquences suivantes :* $S_1 =<$ *(a,c) (e) (m,n)* $>$, $S_2 =<$ *(a,d) (e) (h) (m,n)* $>$, $S_3 =<$ *(a,b) (e) (i,j) (m)* $>$ *et* $S_4 =<$ *(b) (e) (h,i) (m)* $>$. *Les étapes conduisant*

Etape 1 :				
S_1 :	<(a,c)	(e)	()	(m,n)>
S_2 :	<(a,d)	(e)	(h)	(m,n)>
SA_{12} :	(a :2, c :1, d :1) :2	(e :2) :2	(h :1) :1	(m :2, n :2) :2
Etape 2 :				
SA_{12} :	(a :2, c :1, d :1) :2	(e :2) :2	(h :1) :1	(m :2, n :2) :2
S_3 :	<(a,b)	(e)	(i,j)	(m)>
SA_{13} :	(a :3, b :1, c :1, d :1) :3	(e :3) :3	(h :1, i :1, j :1) :2	(m :3, n :2) :3
Etape 3 :				
SA_{13} :	(a :3, b :1, c :1, d :1) :3	(e :3) :3	(h :1, i :1, j :1) :2	(m :3, n :2) :3
S_4 :	<(b)	(e)	(h,i)	(m)>
SA_{14} :	(a :3, b :2, c :1, d :1) :4	(e :4) :4	(h :2, i :2, j :1) :3	(m :4, n :2) :4

FIG. 2 – *Etapes de l'alignement de séquences*

à l'alignement de ces séquences sont détaillées dans la figure 2. Tout d'abord, un itemset vide est inséré dans S_1. Ensuite S_1 et S_2 sont alignées dans le but de produire SA_{12}. Le processus d'alignement est alors appliqué entre SA_{12} et S_3. La méthode d'alignement continue à traiter les séquences deux par deux jusqu'à la dernière séquence.

À la fin du processus d'alignement, la séquence alignée (SA_{14} dans la figure 2) est considérée comme le centroïde du cluster.

4.2 Comparaison séquence/centroïde

Soit s la séquence à affecter dans un cluster et C l'ensemble des clusters. ICDS parcourt l'ensemble des clusters de C et pour chaque cluster $c \in C$, effectue une comparaison entre s et ς_c (le centroïde de c, qui est donc un alignement). Cette comparaison est basée sur la plus longue sous-séquence commune (PLSC) entre s et ς_c. Ensuite, la longueur de la séquence est également prise en compte car elle doit être comprise entre 80% et 120% de la longueur de la séquence alignée.

Soit t la longueur de la première séquence insérée dans c. Les conditions pour que s soit affectée à c sont donc les suivantes :

- $\forall d \in C / d \neq c, dist(s, \varsigma_c) \leq dist(s, \varsigma_d)$
- $0.8 \times t \leq |s| \leq 1.2 \times t$
- $dist(s, \varsigma_c) < 0.3$

La première condition assure que s est affectée dans le cluster dont le centroïde est le plus similaire à s. La deuxième condition assure que les clusters contiendront des séquences de taille homogène et que la taille moyenne des séquences d'un cluster variera peu. Enfin la troisième condition assure que si aucun centroïde ayant un degré de similitude supérieur à 70% avec s n'est trouvé, alors s n'est affectée à aucun cluster. Dans ce dernier cas, un nouveau cluster est créé et s y est affectée.

4.3 Aspect incrémental

Dans ICDS, nous avons implémenté une segmentation incrémentale des séquences (ce qui la rend semi-supervisée) comme illustré à la figure 3. La segmentation commence au batch B_0 pour lequel aucune information n'est disponible et nous obtenons un découpage du batch en plusieurs clusters. Ce découpage est conservé lors du passage au batch B_1. Chaque séquence de B_1 est alors comparée aux centroïdes des clusters obtenus sur le batch B_0. A la fin du traitement de B_n (le n^{eme} batch), la segmentation peut avoir évolué par rapport à celui de B_{n-1} selon trois cas possibles :

1. Modification du centroïde d'un cluster. Dans ce cas, le contenu du cluster a changé et son centroïde s'est déplacé (l'alignement est différent avec des éléments nouveaux, des poids différents, une modification de leur ordonnancement, etc.).
2. Disparition de clusters. Dans le cas où un cluster deviendrait trop petit pour être pris en compte, ou bien si il n'accueille aucune séquence, il disparaît.
3. Apparition de clusters. Un nouveau cluster apparaît, signe d'une évolution importante des comportements.

Et ce traitement continue, en faisant évoluer la segmentation au fil du flux. Le pseudo-code d'ICDS est donnée ci-dessous :

Algorithm (ICDS)
Input : $B = \cup_{i=0}^{\infty} B_i$: un ensemble infini de batches de transactions ; $minSize$: la taille minimum des classes à synthétiser ; $minSim$: la similitude minimum entre deux séquences pour qu'un cluster soit mis à jour ; k : le filtre dans la méthode d'alignement.
Output : L'arbre préfixé mis à jour avec les séquences alignées.
$c \leftarrow \oslash$;
While (B) **Do**
  $b \leftarrow$ NextBatch() ;
  $C \leftarrow$ Clustering(c, b, $minSize$, $minSim$) ; // 1) Obtenir des cluster de taille $> minSize$ // et marquer les clusters non utilisés
  Foreach ($c \in C$) **Do** // 2) Résumer chaque cluster avec un filtre k ;
    If (notUsed(c)) **Then** Delete(c) ; // Disparition de clusters
    Else DeleteOldAlignment(c) ; // Apparition ou évolution de clusters
      $SA_c \leftarrow$ Alignment(c, k) ;
      PrefixTree $\leftarrow$ PrefixTree+SA_c ; //3) Stocker les séquences alignées
  Done
Done (fin Algorithm ICDS) ;

Au début de l'algorithme, C, le résultat de la segmentation, contient un ensemble vide de clusters. Cet ensemble est alors mis à jour par la méthode "Clustering(c,b, $minSize$, $minSim$)" qui prend comme variables d'entrée c, le résultat de la segmentation au batch précédent (au début de l'algorithme, il n'y a pas eu de batch précédent, donc le paramètre c est ignoré) ; b, les séquences du batch ; $minSize$,la taille minimum d'un cluster à conserver en fin de traitement ; et enfin $minSim$, la similitude minimum pour affecter une séquences dans un cluster.
Pour les batches suivants, le traitement est similaire et pour chaque appel à la méthode "Clustering" le résultat de la segmentation sur le batch précédent est pris en compte.

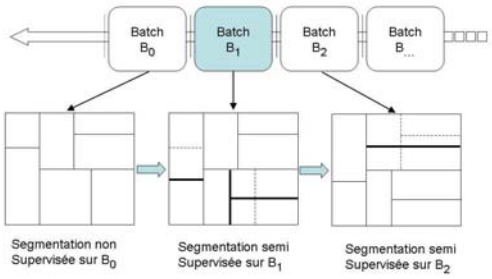

FIG. 3 – *Principe de la segmentation incrémentale dans un flux de données*

5 Expérimentations

ICDS a été implémenté en Java sur un Pentium (2,1 Ghz) exploité par un système Linux Fedora. Nous avons évalué notre proposition sur des données réelles issues des usages du Web de l'Inria Sophia Antipolis.

5.1 Temps de réponse comparés de ICDS et SCDS

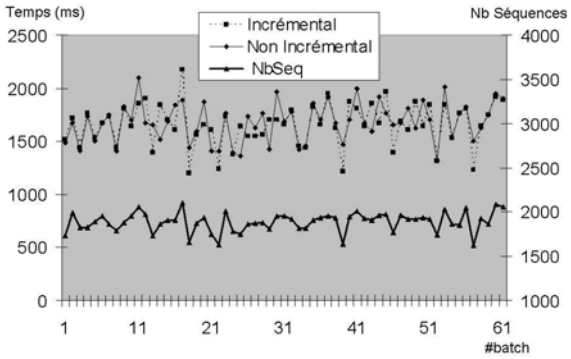

FIG. 4 – *Temps d'exécution de SCDS.*

Dans le but de comparer les temps de réponse d'ICDS de SCDS, nous reportons à la figure 4 le temps nécessaire pour classer les séquences sur chaque batch correspondant à des données d'usage sur le site Web de l'Inria. Les données ont été collectées sur une période de 14 mois et représentent 14 Go. Le nombre total de navigations est de 3,5 millions pour 300000 navigations. Nous avons découpé le fichier log en batches de 4500 transactions (soit environ 1500 séquences en moyenne). Nous pouvons observer que les temps de réponse sont assez semblables (ils varient de 1200 ms à 2200 ms) avec un avantage tangible pour SCDS. En effet, le

temps moyen d'exécution de ICDS est supérieur de 5% à celui de SCDS. Cela peut s'expliquer de deux manières :

1. Au début du traitement d'un batch B_n, soit dès le traitement de s_1 (la première séquence de B_n) il faut parcourir un nombre déjà grand de clusters existants (les clusters hérités du batch B_{n-1}) afin de trouver celui qui pourrait correspondre à s_1. Alors que dans SCDS, l'ensemble des clusters est vide à l'initialisation d'un batch, donc s_1 n'est comparée à aucun autre cluster et un premier cluster est créé.
2. A la fin du traitement du batch il faut recalculer les alignements pour ne plus tenir compte des anciennes séquences de chaque cluster.

Nous avons ajouté à la figure 4 le nombre de séquences de chaque batch pour expliquer les différences de temps d'exécution d'un batch à un autre. On peut observer, par exemple, que le batch 1 contient 1750 séquences. Il s'agit d'un premier élément montrant qu'il n'est pas justifié de passer d'une méthode amnésique à une méthode incrémentale dans notre cas.

5.2 Analyse de la qualité des clusters

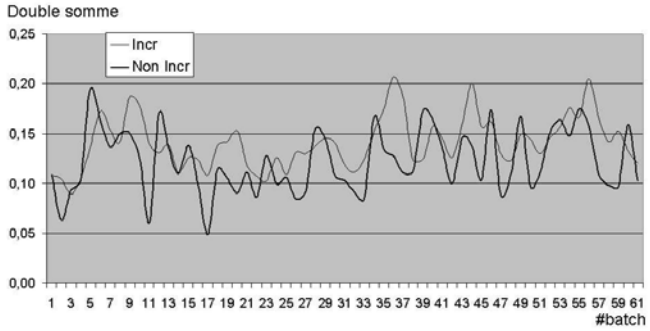

FIG. 5 – *Distance globale, batch par batch*

Afin de mesurer la qualité des classes produites par ICDS, notre principal outil sera la distance entre deux séquences. Soit s_1 et s_2, deux séquences, la distance $dist(s_1, s_2)$ entre s_1 et s_2 est basée sur $sim(s_1, s_2)$, la mesure de similitude donnée par la définition 3 et telle que $dist(s_1, s_2) = 1 - sim(s_1, s_2)$. On a donc $dist(s_1, s_2) \in [0..1]$ et $dist(s_1, s_2)$ proche de 0 signifie que les séquences sont proches (similaires si cette valeur est nulle) alors que $dist(s_1, s_2)$ proche de 1 signifie que les séquences sont éloignées (ne partagent aucun item si cette valeur est 1). Nous reportons dans la figure 5 la double moyenne DBM après avoir traité chaque batch. Soit C l'ensemble des classes, DBM est calculée de la manière suivante :

$$DBM = \frac{\sum_{i \in C} \frac{\sum_{x \in C_i} dist(x, c_i)}{|C_i|}}{|C|}$$

Avec c_i le centre de C_i (la i^{eme} classe). Soit C_i la i^{eme} classe, le centre de C_i est une séquence c_i telle que :

$\forall s \in C_i, \sum_{x \in C_i} dist(s, x) \geq \sum_{y \in C_i} dist(c_i, y)$.

La valeur finale de DBM à la fin du batch est donnée par la figure 5. On peut y observer que DBM est comprise entre 5% et 18%. A la fin du processus, la valeur moyenne de DBM est de 14% (une qualité moyenne des classes de 86%). Pour SCDS cette moyenne est de 12% (soit une qualité de 88%). Une fois de plus, il s'agit d'un élément en faveur de SCDS et ne justifiant pas le passage à l'incrémental.

Nous avons également implémenté une segmentation hiérarchique sur les séquences de chaque batch. Afin de compléter notre étude de la qualité des clusters obtenus avec ICDS, nous avons utilisé les mesures d'entropie et pureté, par comparaison avec les clusters obtenus par la segmentation hiérarchique. L'entropie d'un cluster C de taille n_r est calculée selon la formule suivante : $E(C) = -\frac{1}{logq}\sum_{i=1}^{q}\frac{n_r^i}{n_r}log\frac{n_r^i}{n_r}$, où q est le nombre total de clusters et n_r^i est le nombre de séquences du i^{eme} cluster qui font partie du cluster C. L'entropie de la segmentation est ensuite donnée par la formule :

$Entropie = \sum_{r=1}^{k}\frac{n_r}{n}E(C_r)$, avec n le nombre total de séquences. On considère qu'une petite valeur d'entropie traduit une bonne segmentation (par rapport à la segmentation de référence).

La pureté d'un cluster est donnée par : $P(C_r) = \frac{1}{n_r}max(n_r^i)$ et la pureté de la segmentation est donnée par la formule suivante : $Purete = \sum_{r=1}^{k}\frac{n_r}{n}P(C_r)$. Une grande valeur de pureté traduit une bonne segmentation par rapport à la segmentation de référence.

Les valeurs que nous avons obtenues pour l'entropie et la pureté sur les 10 premiers batches sont données dans le tableau suivant :

Batch	Entropie SCDS	Entropie ICDS	Pureté SCDS	Pureté ICDS
1	0,012523916	0.012523916	0,95107037	0.95107037
2	0,013794152	0.015740575	0,94326377	0.9367343
3	0,017435603	0.017890325	0,93279576	0.9295173
4	0,015584618	0.014704098	0,93603873	0.9371327
5	0,016993482	0.016103497	0,93243974	0.9366611
6	0,018524481	0.018878395	0,92739403	0.9258607
7	0,0202273	0.021773964	0,9281229	0.92114794
8	0,015651526	0.017257141	0,9380881	0.93306845
9	0,017332898	0.015581713	0,93359625	0.9383797
10	0,01842611	0.016277248	0,93085754	0.93546456

Lors de ces expérimentations, pour SCDS la valeur moyenne de l'entropie a été de 0.0166 et la valeur moyenne de la pureté a été de 0.9353. Pour ICDS la valeur moyenne de l'entropie a été de 0,0166 et la valeur moyenne de la pureté a été de 0,9345. On constate donc une changement (non significatif) sur la pureté. Cette troisième évaluation de nos résultats confirme les limites d'une approche incrémentale pour la segmentation des données d'un flux.

6 Conclusion

Dans ce papier, nous avons proposé la méthode ICDS pour classer les séquences dans les flux. Notre algorithme repose sur une technique d'alignement des séquences et une segmentation basée sur une comparaison des séquences avec le centroïde de chaque cluster. Ce centroïde

est représenté par l'alignement calculé sur le cluster de manière incrémentale. De plus notre méthode fonctionne de manière incrémentale en gardant le résultat de la segmentation d'un batch sur l'autre afin de faire évoluer ce résultat au fil des batches. Nos expérimentations ont montré que la méthode de traitement incrémentale, comparée à une méthode non incrémentale, est un peu plus complexe à l'exécution (avec des temps de réponse plus longs). De plus, cette complexité ne se justifie pas forcément avec de meilleurs résultats. Il est reconnu que la clé du traitement des flux de données réside dans un compromis entre la qualité des résultats et le temps d'exécution. Cela doit être mis en balance avec le fait que les méthodes incrémentales sont conçues pour donner un résultat exact. À la lumière de nos expérimentations, l'adaptation de méthodes incrémentales au contexte des flux nous semble donc demander une grande attention avant d'être sérieusement envisagée.

Références

Agrawal, R. et R. Srikant (1995). Mining Sequential Patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*, Taiwan.

Garofalakis, M., J. Gehrke, et R. Rastogi (2002). Querying and mining data streams : you only get one look a tutorial. In *SIGMOD '02 : Proceedings of the 2002 ACM SIGMOD international conference on Management of data*.

Giannella, C., J. Han, J. Pei, X. Yan, et P. Yu (2003). *Mining Frequent Patterns in Data Streams at Multiple Time Granularities*. In H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha (eds.), Next Generation Data Mining. AAAI/MIT.

Hay, B., G. Wets, et K. Vanhoof (2002). Web Usage Mining by Means of Multidimensional Sequence Alignment Method. In *WEBKDD*, pp. 50–65.

Kum, H., J. Pei, W. Wang, et D. Duncan (2003). ApproxMAP : Approximate mining of consensus sequential patterns. In *Proceedings of SIAM Int. Conf. on Data Mining*, San Francisco, CA.

MAIDS. MAIDS project : http ://maids.ncsa.uiuc.edu/index.html.

Marascu, A. et F. Masseglia (2006a). Classification de flots de séquences basée sur une approche centroïde. In *XXIVème Congrès INFORSID Informatique des organisations et systèmes d'information et de décision*, Hammamet, Tunisie.

Marascu, A. et F. Masseglia (2006b). Mining sequential patterns from data streams : a centroid approach. *Journal of Intelligent Information Systems (JIIS). 27*(3), 291–307.

Masseglia, F., F. Cathala, et P. Poncelet (1998). The PSP Approach for Mining Sequential Patterns. In *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery*, Nantes, France.

Raissi, C., P. Poncelet, et M. Teisseire (2005). Need for SPEED : Mining Sequential Pattens in Data Streams. In *Actes des 21iemes Journees Bases de Donnees Avancees (BDA 2005)*.

Srikant, R. et R. Agrawal (1996). Mining Sequential Patterns : Generalizations and Performance Improvements. In *Proceedings of the 5th International Conference on Extending Database Technology (EDBT'96)*, Avignon, France, pp. 3–17.

Teng, W.-G., M.-S. Chen, et P. S. Yu (2003). A Regression-Based Temporal Pattern Mining Scheme for Data Streams. In *VLDB*, pp. 93–104.

MENU-NN: Mining Network Data Streams via NN Clustering

Christopher C. Toombs*, Abdelghani Bellaachia**

Computer Science Department
The George Washington University
Washington, DC 20052, USA
*cctoombs@gwu.edu
**bell@gwu.edu

Abstract. Traditional network intrusion detection systems (NIDS) use signatures of previously seen attacks to determine if currently observed network traffic is malicious or not. More recently, data mining approaches have been applied to the NIDS problem in an attempt to extract patterns of "normal" or acceptable traffic, and "anomalous" or unacceptable traffic. Due to the nature of network communications and security monitoring, NIDS requires a stream mining approach. This paper reviews several previously describes NIDS methods and discusses their successes and failures. We propose the Memory Efficient NIDS Using NN Clustering (MENU-NN) system, which attempts to perform stream-mining NIDS without using training data. This is achieved through a bi-level clustering technique that classifies and discards connections as quickly as possible, preserving only the connections that require a broader context for future classification. This approach seems to improve upon similar previous work.

1 Introduction

A data security practitioner must focus on three sub-tasks to achieve his or her goal: data confidentiality, data integrity, and data availability. There does not yet exist a useful system that can guarantee the permanence of all three of these conditions; there is no system that can be considered to be secure. Because such a system does not exist, and because building one would be excruciatingly difficult and expensive, security professionals cannot effectively prevent attacks on their systems. In light of this, such professionals should desire tools and technologies to detect malicious events, so that they can respond appropriately and minimize the impact of such events.

Despite the name, an *Intrusion Detection System* (IDS) is a tool that can monitor a system for any type of activity, not just intrusions, although they do tend to be used to identify malicious events. The goal of the tool is to passively monitor a system for events it deems noteworthy. Ideally, this would consist solely of behavior that is in violation of the security policies of the system (Brugger, 2004). Once such an event is detected, the IDS will report this behavior, as well as related information such as the time of the event and its source, to a security authority for manual review and response. A *Network IDS* (NIDS) is an IDS designed specifically to monitor the communication links between nodes.

For maximum benefit, a NIDS will be located on a link which carries all traffic for all communication nodes requiring security monitoring. Taking into account current practical limitations on computing power and data storage, a system generating an amount of

communications traffic above a particular threshold must be able to rapidly evaluate the stream of data and provide a decision. With this in mind, security professionals would clearly desire a stream mining approach to NIDS.

A misuse-based IDS, also known as a *Signature-based IDS*, will compare current system behavior with a large set of behaviors (signatures) that are defined to be malicious (Brugger, 2004 & Lazarevic et. al., 2003). These monitors work very well at alerting system owners to possible known intrusion because almost all known intrusions have a unique feature with which we can compare the current behavior of the system. Unfortunately, these types of monitors, being based on patterns of previously seen misuse, are poor at detecting new attacks.

In contrast to the misuse-based IDS, an anomaly-based IDS operates without pre-defined signatures by learning patterns for normal behavior and will label any behavior which falls outside this pattern as unusual/attack. This approach tends to be much more successful than the misuse-based approach at identifying novel attacks which are attacks that have not yet been seen (Lazarevic et. al., 2003 & Leung, 2005). Anomaly-based IDSs can be further bisected: one class which makes use of training data in order to learn patterns of normality prior to operation, and one class which relies solely upon the data it receives during operation. Unfortunately, due to its very nature, all anomaly-based approaches also tend to have higher false-positive rates when compared to misuse-based IDSs (Lazarevic et. al., 2003).

This paper will focus almost exclusively on anomaly-based NIDS methods. As mentioned earlier, such approaches make use of ideas and concepts from data mining and artificial intelligence to learn normal patterns of behavior. First, we will briefly discuss two generic anomaly-based detection approaches. Then, we will review specific implementations of these approaches and discuss the successes and failures of each. We will next propose a new strategy for learning patterns of normal behavior in network and will discuss the results of this approach. Finally, we will offer insight into past approaches, current designs, and the future of novel network intrusion detection.

2 Previous Work

The groundwork for using DM and AI techniques for NIDS began in 1987 with SRI's Intrusion Detection Expert System (IDES). At the time, the methods IDES used were not considered DM or AI, but rather simple statistical methods for determining the unlikely-ness of a new event. While IDES was confined to a local machine, and did not operate on a network, the work inspired others to expand the research onto networks. This section looks at different implementations, beginning after IDES, and examines the progression and evolutions of NIDS.

2.1 eBayes TCP

eBayes TCP (Valdes, 2000) was developed by SRI between 1996 and 1997 as a component in the Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) framework which built on the work of IDES (Porras, 1997). EMERALD includes many of the features from IDES along with signature-based (misuse-based) intrusion detection capabilities and network intrusion detection functions, which is our focus.

In the eBayes TCP system, anomaly-detection decisions are made using a tree structure in which direct observations of the communication are made at the leaves of the tree, and the belief in a particular state is determined at the root.

This implementation has been evaluated using the 1999 DARPA Intrusion Detection Evaluation data sets. The authors claim it is "highly effective against floods and non-stealthy probe attacks, and moderately effective against stealthy probe attacks." Of particular note, they claim to have categorized some connections as OTHER, which turned out to be attacks for which they had no corresponding prior values. If true, this may demonstrate the power of their approach, in that it can, with some degree of accuracy, detect new attacks about which it has no prior knowledge. While eBayes TCP is not a stream-mining approach and does require training data it is an important step in realizing the potential of data mining approaches to NIDS.

2.2 Packet Header Anomaly Detection (PHAD)

PHAD (Mahoney, 2001), from the Florida Institute of Technology, may be the simplest network-based anomaly detection system described in this paper. The goal of PHAD is to use only the header data at various layers of the OSI model, and not look at the data, to determine whether a packet is malicious. The simple assumption made by the creators of PHAD is that if a packet occurs with a probability of p, its anomaly score should be 1/p. To support this observation, the creators have developed a simple technique for calculating the value of p.

PHAD was subjected to the 1999 DARPA Intrusion Detection Evaluation data for analysis. During the detection phase, of the top twenty most anomalous packets, as measured by PHAD, eight corresponded to actual attacks. Nine of the twenty alarms were only false positive while three packets did not meet the criteria for detection despite having an anomalous score (this is because there represented an attack where there was no IP address, and PHAD does not define detection when there is no IP address). By DARPA criteria, PHAD, using C=32, detected 67 of the 201 attack instance, with a false alarm rate of 10 per day. Of attacks it did not detect, many occur at the application layer, which PHAD does not monitor (attacks in HTTP, FTP, DNS, etc…). The best part of PHAD is its run-time overhead in time and space. Due to its simplicity, it processes 2.9GB of training data and 4.0GB of test data in 364 seconds (95,900 packets/second) on a Sparc Ultra 60 with 450 MHz 64-bit processor, 512 MB memory and 4MB cache. There is 23 seconds of overhead per simulated day (0.026% at the simulation rate). The memory usage is 34 fields times 32 pairs of 4-byte integers to represent bounds of each cluster = 8KB total. Although PHAD requires training data, once trained, PHAD could be deployed on a network to analyze live data streams, thanks to its meager memory and processing power requirements, and its packet-by-packet analysis method.

2.3 Intelligent Intrusion Detection System (IIDS)

IIDS completed development in 2000 at Mississippi State University and uses a slightly different approach for mining intrusions than previous implementations, such as fuzzy data mining and genetic algorithms to determine which connections are anomalous and which are not. IIDS has two detection approaches: one is a traditional rule-based expert systems

technique for misuse-based detection, and the other is a fuzzy data mining technique (Bridges, 2000).

Fuzzy membership applies to just one dimension of a connection. A connection will be scored into a class from {“low”, “medium”, “high”} by finding the product of the membership classes for its constituent dimensions. This will result in rules that predict, with some confidence and support, an attribute of an event based on observed attributes of previously occurring events. If the observation is in disagreement with the prediction, this will raise an alert.

IIDS was a prototype system and has not been evaluated against any data sets. While it is difficult to know, at this point, whether this line of research represents a promising direction or not, the concept of analyzing sequences of events (windows) without requiring the complete data set during analysis will be valuable in the construction of an unsupervised stream-mining NIDS.

2.4 Minnesota Intrusion Detection System (MINDS)

MINDS (Erotz et. al., 2003, Erotz et. al. 2004 & Lazarevic et. al., 2003), which completed development at the University of Minnesota in 2003, uses a combination of data mining methods to detect intrusions. MINDS is similar to PHAD because it analyzes on the header information of network traffic, rather than the entire data stream.

The first step in the MINDS system is the extraction of three classes of network features; features that trigger an alarm at this stage are removed from further processing. The creators of MINDS evaluated four outlier detection schemes and an unsupervised support vector machine (USVM) approach. The four outlier detection schemes evaluated were: kth nearest neighbor (kNN), nearest neighbor (NN), Mahalanobis-distance based outlier detection, and density based local outlier (LOF).

The developers of MINDS evaluated their approach on 1998 DARPA Intrusion Detection Evaluation data. For “bursty” attacks, in order of decreasing detection rate, the algorithms fared in this order: NN, LOF, USVM, and Mahalanobis-based approach. Note that kNN is not included in this list, because kNN was not used in this particular evaluation. When comparing the detection rate to the false alarm rate, LOF fares best, followed by NN (Erotz et. al. 2004 & Lazarevic et. al., 2003). For single-connection attacks, LOF fares best, followed by USVM, then NN and finally Mahalanobis-based. Unfortunately, MINDS requires labeled training data and operates in a static environment. However, tthe work on MINDS shows the strength of the NN approach in analyzing network header data.

2.5 fpMAFIA

The details of fpMAFIA (Leung et. al., 2005), were developed at the University of Melbourne. This is the first method this paper discusses which uses an entirely unsupervised detection algorithm. Their approach is based upon grid-based methods of clustering such as CLIQUE and pMAFIA. The developers name four specific challenges they wished to address with fpMAFIA: efficient processing of data, high dimensionality of network data, accurately determine the boundary of clusters, and ensuring the algorithm covers over 95% of the data set.

fpMAFIA is a grid-and-density based clustering algorithm that is similar to pMAFIA. The developers use multi-stage clustering and FP-Trees to achieve the goal of detecting

novel network attacks in a set of network traffic. This approach was evaluated on KDD Cup 1999 data without labels associated with the training data. The developers of fpMAFIA note that the area under the ROC curve for their approach is less than the area under the ROC for the kNN approach, an SVM approach, a fixed-width clustering approach, and a modified clustering-tv approach. Its area under the ROC curve is .867, so while obtaining a high detection rate, it suffered from a high false positive rate as well. Also, as mentioned earlier, this approach is unsupervised, a goal for NIDS, but must analyze the data set in its entirety; this is not a stream-mining approach to NIDS.

While this approach does not require labeled training data, it does require a learning phase. During this phase, clusters are created which function as classifiers for testing data. More interesting, this approach does not give any weight to time as previous methods have. That is, all connections are equally weighted, even if one occurred two years ago in the data set. This may or may not be a valid assumption. The next approach is based on the idea that this is not a valid assumption and that more recent events tend to be of greater relevancy in detecting anomalies.

2.6 CluStream

The details of CluStream (Aggarwal et. al., 2003), from the T.J. Watson Research Center, were originally published in 2003. The driving force behind this approach is that data streams evolve over time, and therefore, clusters should take time into account even when it is not explicitly part of the data being analyzed. To address this problem, the developers suggest a *pyramidal time frame* in conjunction with a *micro-clustering* approach. CluStream achieves these goals by periodically storing detailed summary statistics and, offline, computing user-specified temporal clusters. Clusters may be merged at any time if memory space is at a premium, in which case the distance between cluster centroids will be used to determine the micro-clusters to be merged.

2.7 Enhanced Stream Mining

ESM uses a data preprocessing step, followed by a bi-level clustering algorithm with compressed time analysis to classify incoming network communication data (Bellaachia, et. al., 2003).

During the preprocessing step, each incoming packet is assigned a score based on the cosine multiplication of each dimension by the model's median vector. Packets are then clustered into three categories based on this score: high anomaly, normal, or low anomaly. Immediately after this first clustering step, metadata about each cluster is extracted for further analysis. This information forms what is known as the *Observation Layer* (O). Because we are interested in detection outliers (anomalies), more detailed information about outliers is extracted to make up the data in the *Minimal Interest Layer* (M). Note that this step can occur based solely on the window clustering from the lower layer, which means the packet can be forwarded to its destination while this extraction occurs.

ESM was evaluated against KDD Cup 1999 data. The experimenters created 6 classes of datasets from the original data – four contained attacks of the four classes previously mentioned, while two classes were randomized data sets from the original. The measure used was the number of true negatives and false positives detected. The detection rate (detection of true positives) was very low for all types of attacks, never exceeding 37% for

any class of attack or frame size. This approach seems promising in that it is very efficient and will not unnecessarily slow down the network traffic on the network on which the IDS is operating.

3 Memory Efficient NIDS Using NN Clustering (MENU-NN)

This paper presents an approach similar in nature to ESM, called Memory Efficient NIDS Using NN Clusters (MENU-NN). The goal of this approach is to produce an effective network anomaly detection algorithm that does not make use of training data, nor requires the storage of an entire set of data for proper analysis. Like ESM, MENU-NN is a bi-level clustering algorithm that leverages the success of NN clustering exhibited by MINDS. In short, connections are grouped in windows, which, when full, are clustered using NN, as shown in Figure 1. After clustering, most connections can be classified and disposed, while a subset will saved, in full, for further clustering. Once a pre-determined number of windows have been processed, MENU-NN will cluster all the saved connections over these windows will by NN once more. This final clustering will provide the classification of these connections.

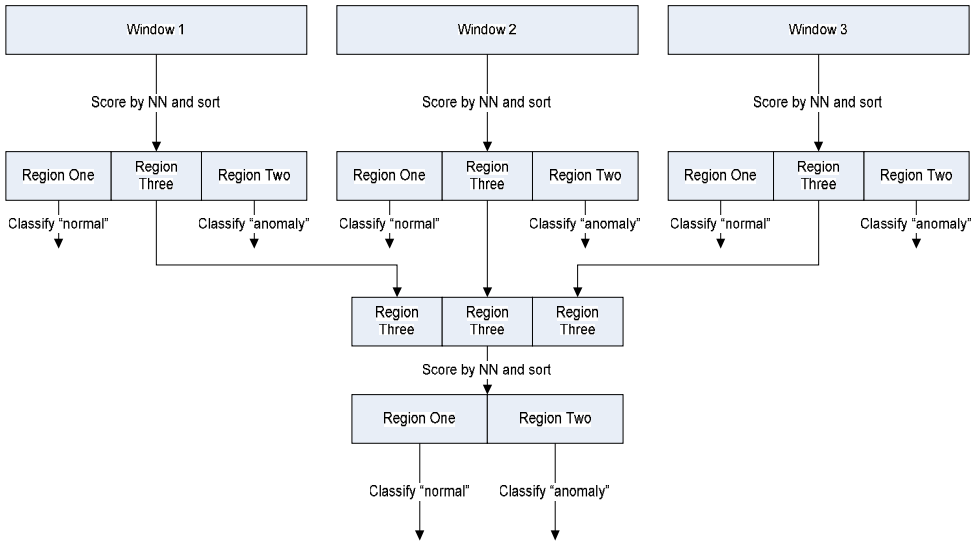

FIG. 1 – *Bi-level NN clustering in MENU-NN.*

When received, data will be initially grouped into windows. A window is a collection of connections. Each window may hold up to W connections, or a window may hold all the connections over T units of time. When the window is full, all connections within the window will be clustered using the NN clustering algorithm previously mentioned. The connections within the window will be sorted by the results of NN.

Clustering connections in the window produces a set of connections with three regions: region one contains connections with low NN values, region two contains connections with

high NN values, while region three has NN values in between regions one and two. Each region is user-defined through parameters to MENU-NN that specify the percentage of connections in a window to include in each region (P_1, P_2, P_3 for regions one, two, and three, respectively). Note that $P_3 = 1 – (P_1+P_2,)$, this is used in actual code – P_3 is not a parameter. Naturally, the sum of these percentages must be 100%. This regionalizing of the connections permits us to immediately classify the connections in region one and region two. These connections can be disposed, providing more memory. The connections from region three may be anomalous or they may be normal; the limited size of the window makes it difficult to classify such connections. For instance, if the vast majority of the connections in the window are normal, then the connections in region three are probably normal as well, and vice-versa if the vast majority are abnormal. Hence, these connections will be preserved and will require a closer look over a larger period of activity.

The user will specify a number of windows R after which region three clustering will take place. After MENU-NN processes R windows, MENU-NN will have R sets of region three connections. MENU-NN will combine the R sets into one set of connections and will calculate the NN distances for each connection. This set will then be sorted on the NN distance for each connection. This set will then be partitioned based on a user-defined parameter F into two sets – one set will be classified as normal connections, while the second set will be classified as abnormal.

MENU-NN uses the merge-sort sorting algorithm for sorting connections based on their NN value. While this is normally an O(nlog(n)) operation, MENU-NN uses a fixed window and region size, which makes each merge-sort operate in constant time, O(k). The number of sorts is still a function of the number of connections, increasing the run-time to O(n). Likewise, the calculation of NN distance is now O(k) rather than $O(n^2)$Analyzing the connections, building connection arrays, and creating regions are similarly efficient. Hence, the overall efficiency, in both time and memory, is O(n).

MENU-NN makes use of all the features found in the KDD Cup 1999 data, except for protocol information and flag information, as these are difficult to categorize when calculating Euclidean distance.

4 MENU-NN Evaluation

MENU-NN has been evaluated using the KDD Cup 1999 Corrected Evaluation Data. The evaluation sought only to determine which connections were normal and which were not. Because the data set does not include a timestamp of each connection, window sizes are based on a total number of connections. This evaluation seeks to evaluate the effect of the window size, and the effect of the number of regions created before clustering Region 3. That is, W and R have been varied or this evaluation, while F, P_1, P_2, P_3 remain constant. W varies from 100 to 500 by increments of 100 connections. R varies from 50 to 250 by increments of 50 iterations. P1 is set at 50% and P2 is set at 30%. Hence, P3 is 20%. Finally, F is set at 60%.

	W	R	True +	False +	True -	False -	Detection Rate
Test One	100	50	31149	163493	86943	29444	51.407%
Test Two	100	100	29105	165568	84868	31488	48.034%
Test Three	100	150	28440	166244	84192	32153	46.936%
Test Four	100	200	27897	166792	83644	32696	46.040%
Test Five	100	250	27642	167050	83386	32951	45.619%
Test Six	200	50	28400	165340	85096	32193	46.870%
Test Seven	200	100	27158	166598	83838	33435	44.820%
Test Eight	200	150	26436	167325	83111	34175	43.616%
Test Nine	200	200	26675	167089	83347	33918	44.023%
Test Ten	200	250	25837	167928	82508	34756	42.640%
Test Eleven	300	50	27357	165902	84534	33056	45.283%
Test Twelve	300	100	26120	167329	83107	34473	43.107%
Test Thirteen	300	150	25608	167845	82541	34985	42.262%
Test Fourteen	300	200	25815	167639	82797	34778	42.604%
Test Fifteen	300	250	25420	168296	82041	35272	41.884%

TAB. 1 – *Summary of the evaluation of MENU-NN against KDD Cup 1999 data, calculated by varying the window size and number of window iterations.*

In meeting its stated goals regarding memory efficiency and scalability in administering judgment on network traffic, MENU-NN performs well. Most importantly, as shown in Table 1, the MENU-NN approach improves upon the detection rate of ESM, by achieving a rate of 51.407% under the best parameters tested – when the window size is 100 and the number of window iterations is 50.

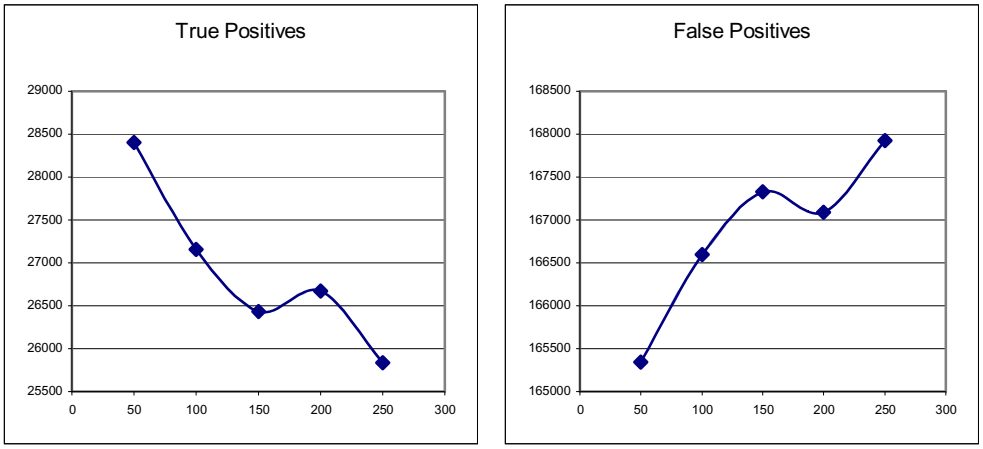

FIG. 2 – *True positives (left) and false positives (right) detected by MENU-NN on KDD Cup '99 data as a function of the number of iterations, when window size is 200.*

As indicated in Figure 2, the most obvious trend from the limited number of simulations performed is that, generally, MENU-NN appears more and more precise and accurate as the number of iterations (R) decreases. This is also true for the window size. However, it should be noted that the number of false positives far outweighs the number of true positives in all cases, which would lead to a saturation of useless alerts in a production system. The current parameters of MENU-NN allow one to specify the percentage of traffic that should be deemed malicious; it is not surprising that the number of false positives is indirectly proportional to the number of true positives.

These trends are reinforced by examining the rate of true negatives and false negatives across the same set of simulations. It remains clear that, for the particular traffic analyzed and for the parameters tested, MENU-NN performed better with a smaller window size and fewer number of iterations.

5 Conclusions and Future Work

While MENU-NN provides a rapid and scalable method for classifying network connections, and does so with a better detection rate than similar previous approaches, there is clearly room for improvement. Most importantly, future work should strive to reduce the number of false positives, as the administrator of such an IDS cannot afford to investigate such a large percentage of incorrect alerts. To do so, it would be prudent to more carefully examine the other parameters of MENU-NN. Fixing these parameters at the moment may have significant effects on the results. First, MENU-NN assumes that within any window there is a percentage of normal traffic and malicious traffic, an assumption that is likely not always correct. The addition of some heuristic against which to evaluate normal traffic and abnormal traffic that would survive across iterations might help solve that problem. This would also require a modification of the method used to determine normal and anomalous connection within any window.

In conjunction with the aforementioned heuristic, using a time-based window, rather than a connection-based window, would likely improve the detection rate. An additional measure analyzing the number of connections within a time-based window could be used to help identify malicious activity, particular Denial of Service attacks.

Finally, it would be useful to make use of protocol and flag information which MENU-NN does not currently examine. Rather than using these directly in calculating Euclidean distance, a table of the frequency of each can be used, and the distance can be evaluated using an inverse frequency function. For instance, if a particular flag exists in 70% of the connections, the corresponding distance value would be 10/7. Correspondingly, a flag that exists in 5% of cases would have a distance of 20.

References

Aggarwal, C. C., J. Han, J. Wang, and P. S. Yu (2003). A framework for clustering evolving data streams. *Proceedings of the 29th VLDB conference*.

Bellaachia, A. and R. Bhatt (2005). An Enhanced Stream Mining Approach for Network Anomaly Detection. *Data Mining, Intrusion Detection, Information Assurance, And Data Networks Security Conference SPIE Conference, Vol. 5812*. Orlando, Florida.

Bridges, Susan M., and Rayford M. Vaughn (2000). Fuzzy Data Mining and Genetic Algorithms Applied to Intrusion Detection. *Proceedings of the Twenty-third National Information Systems Security Conference*. Baltimore, MD.

Brugger S. Terry (2004). Data Mining Methods for Network Intrusion Detection. *Technical Report*. University of California, Davis.

Ertoz, L., E. Eilertson,, A. Lazarevic, P. Tan, P. Dokas, J. Srivastava, and V. Kumar (2003). Detection and Summarization of Novel Network Attacks Using Data Mining. *Technical Report*.

Ertoz, L., E. Eilertson, A. Lazarevic, P. Tan, J. Srivastava, V. Kumar, and P. Dokas (2004). The MINDS – Minnesota Intrusion Detection System, Next *Generation Data Mining*. MIT Press.

Lazarevic, Aleksandar, Aysel Ozgur, Levent Ertoz, Jaideep Srivastava, and Vipin Kumar (2003). A comparative study of anomaly detection schemes in network intrusion detection. *SIAM International Conference on Data Mining*.

Leung, K. and C. Leckie (2005). Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. *Proceedings of the Twenty-Eighth Australasian Computer Science Conference*, 333-342. Estivill-Castro, V., Ed. ACS. Newcastle, Australia.

Mahoney, M. and P. Chan (2001). PHAD: Packet header anomaly detection for identifying hostile network traffic. *Technical Rport CS-2001-4*. Florida Tech.

Porras, P.A. and P.G. Neumann (1997). EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. *Proceedings of the Nineteenth National Computer Security Conference*, 353-365, Baltimore, Maryland.

Valdes A. and K. Skinner (2000). Adaptive, model-based monitoring for cyber attack detection. *In Proceedings of the International Workshop on Recent Advances in Intrusion Detection*.

Classifications non supervisées de données évolutives : application au Web Usage Mining

Alzennyr Da Silva*, Yves Lechavellier*,
Fabrice Rossi*, Francisco De Carvalho**

*Projet AxIS, INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, B.P. 105
78153 Le Chesnay cedex – France
{Alzennyr.Da_Silva, Yves.Lechevallier, Fabrice.Rossi}@inria.fr
**Centro de Informatica - CIn/UFPE
Caixa Postal 7851, CEP 50732-970, Recife (PE) – Brésil
fatc@cin.ufpe.br

Résumé. Il est important pour les opérateurs de sites Web d'analyser l'utilisation de leurs sites afin de mieux connaître le comportement de leurs visiteurs. Il est aussi important de tenir compte de l'aspect temporel dans ces analyses. En effet, la manière dont une visite est réalisée peut changer en raison de modifications liées à la structure et au contenu du site lui-même, ou bien en raison du changement de comportement de certains groupes d'utilisateurs ou de l'émergence de nouveaux comportements. Ainsi, les modèles associés à ces comportements doivent être mis à jour continuellement afin de mieux refléter le comportement actuel des internautes. Une solution, proposée dans cet article, est de mettre à jour ces modèles de comportements à l'aide des résumés obtenus par une approche évolutive des méthodes de classification. Pour ce faire, nous effectuons une partition de la période disponible de temps en sous-périodes de temps plus significatives. Nous comparons les résultats obtenus par cette méthode avec ceux obtenus par une analyse globale.

1 Introduction

Le Web est un des exemples les plus pertinents de source de données volumineuses et dynamiques grâce à l'augmentation colossale du nombre de documents mis en ligne et des nouvelles informations ajoutées chaque jour. Les profils d'accès au Web ont une nature dynamique, dûe au propre dynamisme du contenu et de la structure d'un site Web ou bien dûe au changement d'intérêt de ses utilisateurs au cours du temps. Les profils d'accès à un site Web peuvent être influencés par certains paramètres de nature temporelle, comme par exemple : l'heure et le jour de la semaine, des événements saisonniers (vacances d'été, d'hiver, Noël, etc.), des événements externes dans le monde (épidémies, guerres, crises économiques, coupe du monde de *football*), etc.

Une méthode d'analyse dirigée par l'usage consiste à étudier les traces laissées par les utilisateurs d'un site Web lors de leurs passages, plus précisément les informations enregis-

trées dans des fichiers logs du serveur Web. Dans ce cadre, la plupart des méthodes consacrées à la fouille de données d'usage du Web prennent en compte dans leur analyse toute la période qui enregistre les traces d'usage : les résultats obtenus sont donc naturellement ceux qui prédominent sur la totalité de la période. Ainsi, certains types de comportements, qui ont lieu pendant de courtes sous-périodes ne sont pas pris en compte et restent donc ignorés par les méthodes classiques. Il est pourtant important d'étudier ces comportements et donc de réaliser une analyse portant sur des sous périodes significatives. Ceci permet, par exemple, d'étudier d'éventuels changements des centres intérêts des utilisateurs d'un site Web sur ces sous-périodes de temps. On peut ainsi étudier l'évolution temporelle des profils des utilisateurs en les caractérisant par des descriptions capables d'intégrer l'aspect temporel. Le volume des données considérées étant très élevé, il est en outre important de recourir à des résumés pour représenter les profils considérés.

Toutes ces considérations ont motivé d'importants efforts dans l'analyse de données, notamment pour adapter des méthodes de la fouille de données statiques aux données qui évoluent pendant le temps. Le présent article se place dans ce courant de recherche et propose de suivre le changement de comportement à l'aide des résumés obtenus par une approche évolutive de la classification appliquée sur des sous-périodes de temps.

L'article est organisé comme suit : la section 2 contient un bref état de l'art concernant la problématique abordée, dans la section 3 nous présentons les données d'usage et le site Web qui nous sert de référence. La section 4 décrit l'approche proposée pour l'analyse de l'usage basée en sous-périodes temporelles. Nous présentons aussi dans cette section les expériences réalisées, en analysant les résultats et en les comparant à ceux des méthodes de référence. La dernière section présente les conclusions et les travaux futurs envisagés.

2 Etat de l'art

La fouille du Web (Web Mining, WM) (Kosala et Blockeel, 2000) s'est développée à la fin des années 90 et consiste à utiliser l'ensemble des techniques de la fouille de données afin de développer des outils permettant l'extraction d'informations pertinentes à partir de données du Web (documents, traces d'interactions, structure des pages, structure des liens, etc.). Plus précisément, la fouille de données d'usage du Web (Web Usage Mining, WUM) désigne l'ensemble de techniques basées sur la fouille de données pour analyser le comportement des utilisateurs d'un site Web (Cooley et al., 1999) (Spiliopoulou, 1999) et c'est précisément sur cette partie que le présent article se concentre.

L'analyse de l'usage a commencé relativement récemment à tenir compte de la dépendance temporelle des profils de comportement. Dans (Roddick et Spiliopoulou, 2002), les auteurs examinent les travaux antérieurs. Ils résument les solutions proposées et les problèmes en suspens dans l'exploitation de données temporelles, au travers d'une discussion sur les règles temporelles et leur sémantique, mais aussi par l'investigation de la convergence entre la fouille de données et de la sémantique temporelle. Tout récemment, dans (Laxman et Sastry, 2006) les auteurs discutent en quelques lignes des méthodes pour découvrir les modèles séquentiels, les motifs fréquents et les modèles périodiques partiels dans les flux de données séquentiels. Ils évoquent également des techniques concernant l'analyse statistique de telles approches.

Cependant, la plupart des méthodes d'analyse de l'usage sont appliquées sur la totalité de données disponibles. Ainsi, les changements intéressants qui pourraient se produire dans la

période considérée ne sont pas pris en compte. Par exemple, quand les données proviennent d'une accumulation portant sur une période de temps potentiellement longue (comme dans le cas des fichiers log Web), on s'attend à ce que les comportements évoluent avec le temps. Pour traiter cela, les modèles d'usage découverts doivent être mis à jour continuellement (avec des algorithmes efficaces) afin de suivre le changement de comportement des visiteurs. Ceci exige de la méthode une surveillance continue des modèles découverts, pré-requis d'importance essentielle pour les applications centrées sur la dimension temporelle. Une solution possible pour traiter ce problème est proposée dans cet article : effectuer le partitionnement du temps avant l'application de la méthode et intégrer un suivi temporel des profils de comportement dans l'algorithme d'extraction de ces derniers.

3 Données d'usage

Les données d'usage d'un site Web proviennent essentiellement des fichiers log des serveurs concernés (dans une approche centrée serveur). Chaque ligne du fichier log décrit une requête reçue par le serveur associé : elle indique ainsi le document demandé, la provenance de la requête, la date de la demande, etc. Diverses techniques de pré-traitement permettent d'extraire des logs des *navigations*, comme par exemple celles de Tanasa et Trousse (2004), utilisées dans le présent article. Une navigation est une suite de requêtes provenant d'un même utilisateur et séparées au plus de 30 minutes. Elle constitue donc la trajectoire d'un utilisateur sur le site.

Pour l'application de notre approche, nous utilisons comme site de référence celui du Centre d'Informatique de Recife-Brésil [1], le laboratoire d'un des auteurs. Ce site est constitué d'un ensemble de pages statiques (pages personnelles des professeurs, pages de support de cours, etc.) et dynamiques, ces dernières étant gérées par *servlets* programmées en Java. La partie dynamique du site consiste en 91 pages très bien organisées sous la forme d'un arbre sémantiquement structuré (cf Rossi et al. (2006a) Rossi et al. (2006b) Da Silva et al. (2006a) Da Silva et al. (2006b) pour une analyse de cette partie du site). Nous avons étudié les accès au site du 1 juillet 2002 au 31 mai 2003 (le fichier de logs contient environ 2Go de données brutes).

Afin d'analyser les traces d'usage plus représentatif, nous avons sélectionné les navigations longues (contenant au moins 10 requêtes et avec une durée totale d'au moins 60 secondes) et supposées humaines (le ratio durée sur requêtes doit être supérieur à 4, c'est-à-dire, correspondre à moins de 15 requêtes par minute). Ces efforts ont comme but d'extraire les navigations humaines et d'exclure celles provenant des robots. L'élimination des navigations courtes est motivée par la recherche de patrons d'utilisation du site, à l'exclusion des accès simples (par l'intermédiaire d'un moteur de recherche) qui n'engendre pas un parcours sur le site. Après le filtrage et élimination des cas aberrants nous avons obtenu un total de 138 536 navigations, 184 275 pages (dont 91 dynamiques), 56 314 utilisateurs et 1,19 minutes comme durée moyenne de visualisation de pages.

[1] Le site analysé est actuellement accessible à l'adresse : `http://www.cin.ufpe.br/`

4 Approche de classification par sous-périodes de temps

La caractérisation de groupes d'utilisateurs consiste à identifier des traits d'usage partagés par un nombre suffisant d'utilisateurs d'un site Web et ainsi fournir des indices permettant d'inférer le profil de chaque groupe (Chi et al., 2002) (Da Silva et al., 2006a,b).

L'approche proposée dans cet article consiste dans un premier temps à diviser la période analysée en sous périodes (mois) dans le but de rechercher l'apparition et/ou l'évolution des comportements qu'on manquerait par une analyse globale de toute la période. Ensuite, une classification est réalisée sur les données de chaque sous-période, aussi bien que sur la période complète. Les résultats fournis pour chaque classification sont donc comparés les uns avec les autres. Notons que le partitionnement temporel porte sur l'ensemble des navigations qui est ainsi découpé en sous-groupes. Chaque navigation est donc affectée à un sous-groupe, ce qui est assez différent des analyses statistiques élémentaires dans lesquelles on compte par exemple les accès à une page en fonction de l'heure d'accès, sans tenir compte des navigations.

Dans notre approche, le partitionnement du temps est formulé en fonction du mois, cependant d'autres possibilités de partitionnement du temps (intervalles de 15 jours, jours fériés et non fériés, périodes dans la journée : matin, après midi, soir, etc.) sont aussi envisageables.

Dans un contexte non supervisé, nous avons réalisé quatre classifications :

- **Classification globale** : on réalise une classification sur tout l'ensemble d'individus (navigations), sans tenir compte des sous-périodes de temps. Pour des raisons d'analyse des résultats, on réalise en suite une intersection entre les classes obtenues et la partition temporelle. La classification globale engendre donc une classification chacun des 11 groupes temporels ;
- **Classification locale indépendante** : pour chaque mois analysé, on réalise une classification sur l'ensemble de navigations appartenant au mois en question. Chaque classification est donc indépendante des autres classifications réalisées sur les autres mois ;
- **Classification locale "précédente"** : cette classification peut être obtenue à partir d'une autre classification quand l'algorithme utilisé est capable d'affecter de nouveaux individus aux groupes obtenus (cf la section suivante). On utilise donc la structure classificatoire de la période temporelle précédente pour obtenir une partition sur la période suivante, ce qui correspond dans notre cas à la première étape de l'algorithme de nuées dynamiques ;
- **Classification locale dépendante** : cette classification peut être obtenue avec des algorithmes itératifs de type nuées dynamiques (cf la section suivante) qu'on peut initialiser de façon adaptée. Ici, on initialise la classification d'une période temporelle avec les prototypes obtenus de la classification de la période temporelle immédiatement précédente.

4.1 Algorithme et critères d'évaluation

Pour la classification des navigations dans notre méthode, nous utilisons un algorithme de type nuées dynamiques (cf Celeux et al. (1989)) applicable sur un tableau de données caractérisé par des attributs descriptifs des navigations (voir tableau 1). D'autres algorithmes de partitionnement sont bien entendu envisageables, mais ils doivent être compatibles avec les techniques de classification décrites dans la section précédente. Il faut en particulier :

1. pouvoir affecter de nouvelles observations à une classification existante ;

2. pouvoir initialiser l'algorithme avec les résultats d'une autre réalisation de lui-même.

Pour toutes les procédures de classification, nous avons demandé 10 classes avec un nombre d'initialisations aléatoires égal à 100, sauf dans le cas où l'algorithme était initialisé avec les résultats obtenus de la sous-période temporelle précédente.

No	Champ	Signification
1	IDNavigation	Code de la navigation
2	NbRequests_OK	Nombre de requêtes réussies (statut = 200) dans la navigation
3	NbRequests_bad	Nombre de requêtes échouées (statut $\neq$ 200) dans la navigation
4	PRequests_OK	Pourcentage de requêtes réussies (= NbRequests_OK/ NbRequests)
5	NbRepetitions	Pourcentage de requêtes répétées dans la navigation
6	PRepetitions	Pourcentage de repetitions (= NbRepetitions / NbRequests)
7	DureeTotale	Durée totale de la navigation (en secondes)
8	MDuree	Moyenne de la durée des requêtes (= DureeTotale / NbRequests)
9	MDuree_OK	Moyenne de la durée des requêtes réussies (= DureeTotale_OK / NbRequests_OK)
10	NbRequests_Sem	Nombre de requêtes rapportées aux pages dynamiques qui forment la structure sémantique du site
11	PRequests_Sem	Pourcentage des requêtes sémantiques (=NbRequests_Sem/ NbRequests) dans la navigation
12	TotalSize	Somme d'octets transférés dans la navigation
13	MSize	Moyenne d'octets transférés (= TotalSize / NbRequests_OK)
14	DureeMax_OK	Durée maximale parmi les requêtes réussies

TAB. 1 – *Attributs descriptifs des navigations.*

Pour analyser les résultats, nous utilisons deux critères. Pour une analyse classe par classe, nous considérons la F-mesure de van Rijsbergen (1979). Pour comparer deux partitions, nous cherchons la meilleure représentation de la classe a de la première partition par une classe b dans la seconde partition, au sens de la F-mesure (ce qui nous donne autant de valeurs numériques qu'il y a de classes dans la première partition). Cette mesure permet une analyse fine, mais elle ne tient pas compte des effectifs relatifs des classes. Pour une analyse plus globale, nous utilisons l'indice de Rand corrigé (cf Hubert et Arabie (1985)) qui permet de comparer directement deux partitions. Pour les deux indices, la valeur 0 correspond à une absence totale de liaison entre les partitions considérées, alors que la valeur 1 indique une liaison parfaite.

4.2 Résultats et discussion

Pour mieux comprendre l'évolution des classes par rapport aux sous-périodes de temps analysées, nous avons réalisé un suivi des prototypes des classes (mois par mois) pour la classification locale indépendante et la classification locale dépendante, puis nous avons projeté ces prototypes dans le plan factoriel obtenu sur la population totale (voir figure 1). Sur cette représentation, chaque disque représente un prototype. Dans la classification dépendante (à droite), les dix classes sont représentées par des couleurs différentes et les traits représentent la trajectoire des prototypes. On note une certaine stabilité malgré la diversité de mois analysés. Dans le cas de la classification indépendante (à gauche), la trajectoire temporelle est simplement matérialisée par les lignes qui joignent un prototype à son plus proche voisin dans la période temporelle précédente. Cela ne donne pas des trajectoires parfaitement identifiées car certains prototypes partagent à un moment donné le même prédécesseur. On note en fait que seules quatre classes sont parfaitement identifiées et stables, les autres subissant des fusions et séparations au cours du temps.

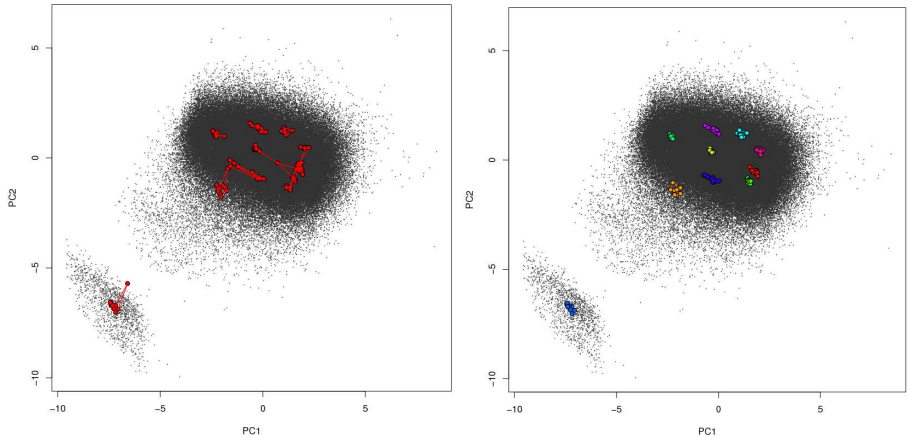

FIG. 1 – *Projection et suivi des prototypes des classes pour les classifications locales : indépendante (gauche) et dépendante (droite).*

Si on projette sur le plan factoriel les prototypes de la classification globale (G1, G2, ..., G10), puis les prototypes obtenus par les classifications locales indépendante et dépendante (voir figure 2), on voit que la classification locale indépendante est capable d'identifier des nouvelles classes qui ne sont pas trouvées par les deux autres classifications.

Nous avons également calculé la variance intra-classe pour les trois classifications (dépendante, indépendante et globale) mois par mois. Comme attendu, le score est meilleur pour la classification locale indépendante, puis classification locale dépendante et finalement pour la classification globale (voir figure 3). A partir de cela, nous pouvons constater que les classes obtenues par la classification locale dépendante présentent plus de cohésion.

Vue la faible différence entre les scores, on s'attendrait donc à ce que les classes soient assez proches. Cependant, la représentation dans le plan factoriel induit un premier doute, car les prototypes dans le cas indépendant semblent parfois assez différents de ceux des cas globaux et dépendants.

En fait, les valeurs de l'indice de Rand ne montrent que la classification indépendante est parfois très différente de la classification globale (figure 4). Ces différences sont confirmées par la F-mesure. Comme on obtient 10 valeurs de F-mesure pour chaque mois (une par classe), on trace une *boxplot* de résumés, ce qui donne la figure 5. On voit que dans le cas de confrontation des classifications indépendante versus globale il y a presque toujours systématiquement des valeurs faibles, c'est-à-dire que certaines classes de la classification indépendante ne sont pas retrouvées dans la classification globale. On voit aussi que la classification "précédente" ne donne pas des résultats très différents de ceux obtenus par la classification dépendante, ce qui confirme l'intuition acquise par l'observation des prototypes dans le plan factoriel : ces derniers bougent "peu" au cours du temps.

En confrontant la classification globale via la F-mesure (classe par classe) aux classifications locales dépendante et indépendante, on affine l'analyse ci-dessus. Ce qui apparaît nettement, c'est que les classes sont très stables dans le temps si on utilise la méthode de classifica-

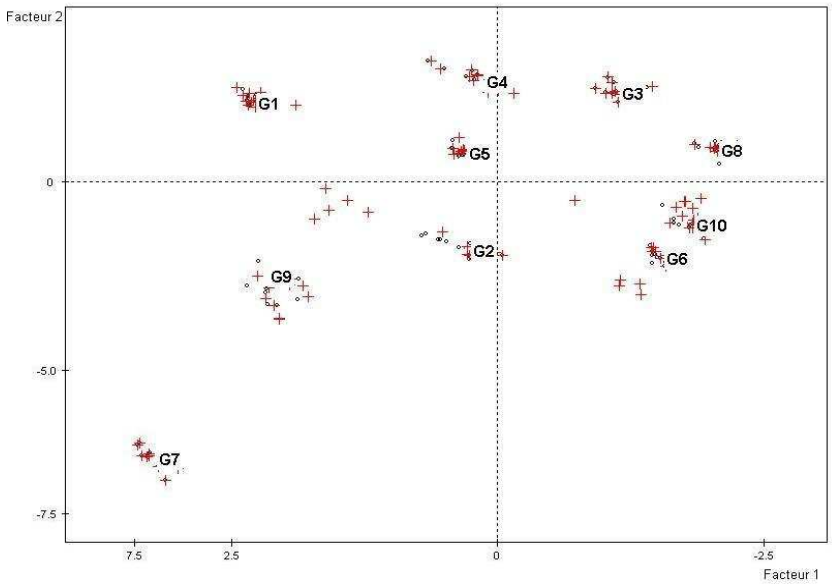

FIG. 2 – *Projection des prototypes des classes pour les classifications : globale (G1, G2, ..., G10), indépendante (+) et dépendante (o).*

tion dépendante. En fait, aucune classe ne descend au dessous de 0.877 pour la F-mesure, ce qui représente une très bonne valeur. Par contre, dans le cas de la classification indépendante, on obtient au contraire des classes très différentes de celles obtenues globalement (avec des valeurs faibles de la F-measures, inférieures à 0.5).

D'une certaine façon, il est surprenant de constater que les partitions découvertes par la classification locale dépendante sont très proches de celles obtenues par la classification globale. Nous pourrions ainsi spéculer qu'une analyse effectuée sur des sous-périodes de temps soit capable d'obtenir les mêmes résultats censés être révélés par une analyse globale effectuée sur toute la période disponible de temps. En outre, la méthode de classification locale dépendante peut être considérée comme une approche de type "diviser pour régner" et pour cela, serait capable de régler certaines contraintes liées au temps total de traitement de données et aux limitations physiques des machines (telles que la taille de la mémoire, vitesse du microprocesseur, etc.).

En conclusion, nous pouvons dire que la méthode de classification locale dépendante montre que les classifications obtenues ne changent pas ou peu au cours du temps (ce qui est confirmé par la variance intra-classe), alors que la méthode de classification locale indépendante ne réussit pas à découvrir ce fait, étant plus sensible aux changements qui peuvent se passer d'une sous-période à l'autre.

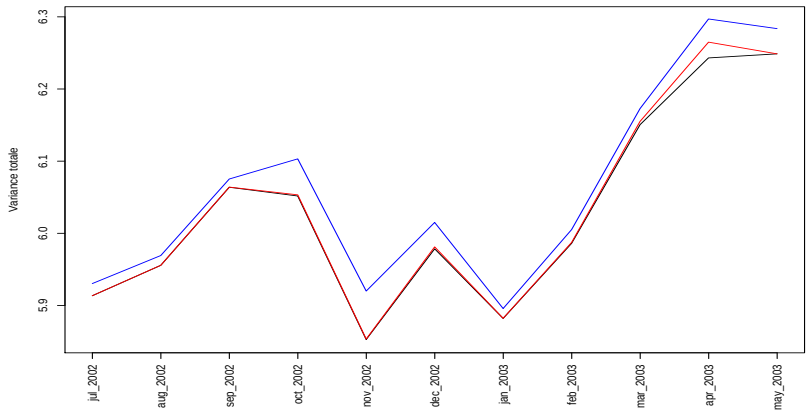

FIG. 3 – *Variance totale pour les classifications : indépendante (trait noir), dépendante (trait rouge) et globale (trait bleu).*

5 Conclusions et perspectives futures

Dans cet article, nous avons abordé la problématique du traitement des données dynamiques dans le contexte de l'analyse de l'usage du Web. Les questions traitées ont montré la nécessité de définition et/ou d'adaptation de méthodes capables d'extraire des connaissances et de suivre l'évolution de ce type de données. Bien qu'il existe de nombreuses méthodes performantes d'extraction de connaissances, peu de travaux ont été consacrés à la problématique de données pouvant évoluer avec le temps.

A travers nos expérimentations, nous avons montré que l'analyse des données dynamiques par sous-périodes offre un certain nombre d'avantages. Ainsi, elle permet de rendre la méthode plus efficace dans la découverte des classes et des possibles changements qui peuvent avoir lieu pendant de courts sous-périodes de temps et ne sont pas détectés par une analyse globale. Dans un plan secondaire, notre approche permet aussi de s'affranchir des difficultés liées aux limites des machines (telles que la taille de la mémoire, vitesse du processeur, etc.) car nous concentrons l'analyse sur une partie des données disponibles.

Comme possibilité de futurs travaux nous pouvons signaler l'application d'autres méthodes de classification et la mise en oeuvre des techniques permettant la découverte automatique du nombre de classes, ce qui permettrait d'identifier plus clairement les possibles fusionnements et scissions des classes au cours du temps.

Remerciements

Nous tenons à remercier le projet INRIA/FACEPE et la CAPES-Brésil pour leur soutien à ce travail de recherche.

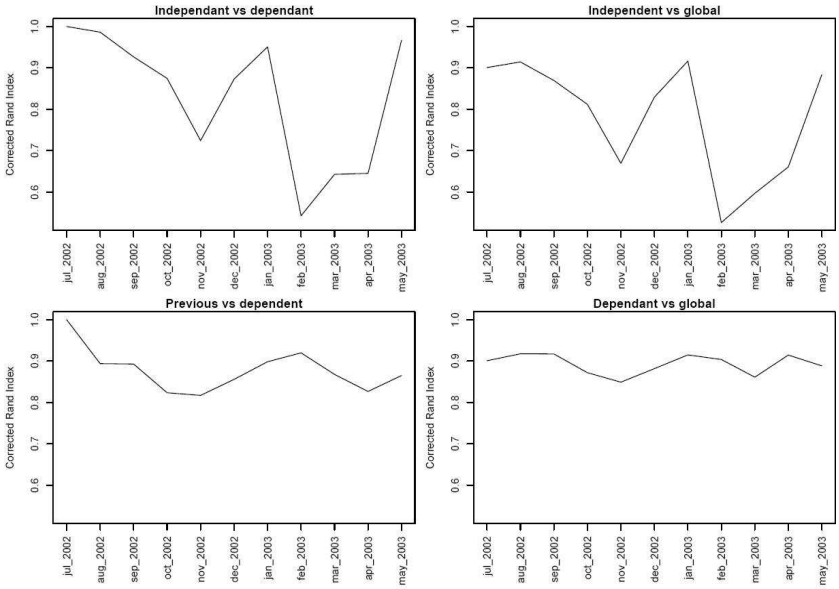

FIG. 4 – *Indice de Rand corrigé classe par classe.*

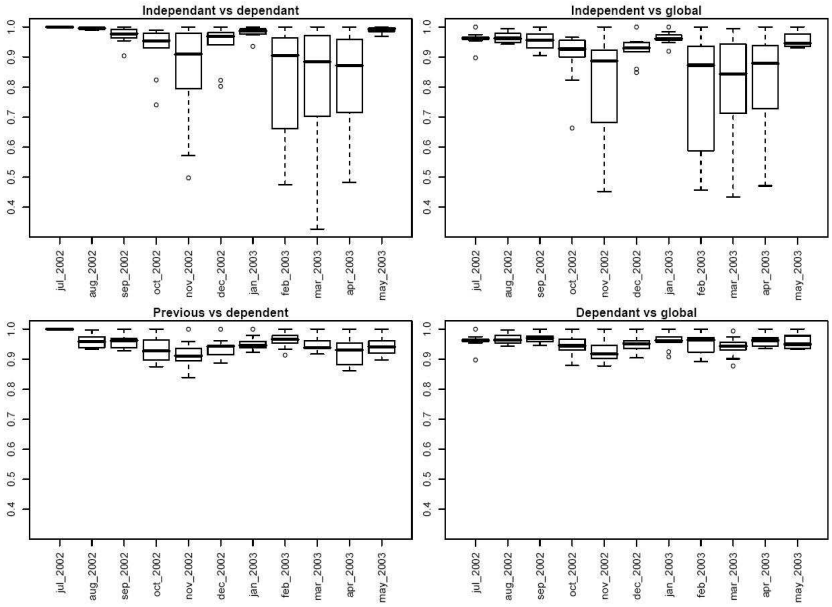

FIG. 5 – *Boxplots correspondant aux F-measures classe par classe.*

Références

Celeux, G., E. Diday, G. Govaert, Y. Lechevallier, et H. Ralambondrainy (1989). *Classification Automatique des Données*. Paris : Bordas.

Chi, E. H., A. Rosien, et J. Heer (2002). Lumberjack: Intelligent discovery and analysis of web user traffic composition. *ACM SIGKDD Workshop on Web Mining for Usage Patterns and User Profiles (WEBKDD)*, 1–16.

Cooley, R., B. Mobasher, et J. Srivastava (1999). Data preparation for mining world wide web browsing patterns. *Journal of Knowledge and Information Systems 1*(1), 5–32.

Da Silva, A., F. D. Carvalho, Y. Lechevallier, et B. Trousse (2006a). Mining web usage data for discovering navigation clusters. *ISCC 2006*, 910–915.

Da Silva, A., F. De Carvalho, Y. Lechevallier, et B. Trousse (2006b). Characterizing visitor groups from web data streams. *GrC 2006*, 389–392.

Hubert, L. et P. Arabie (1985). Comparing partitions. *Journal of Classification 2*, 193–218.

Kosala, R. et H. Blockeel (2000). Web mining research: A survey. *ACM SIGKDD Explorations 2*, 1–15.

Laxman, S. et P. S. Sastry (2006). A survey of temporal data mining. *SADHANA - Academy Proceedings in Engineering Sciences, Indian Academy of Sciences 31*(2), 173–198.

Roddick, J. F. et M. Spiliopoulou (2002). A survey of temporal knowledge discovery paradigms and methods. *IEEE Transactions on KDE 14*(4), 750–767.

Rossi, F., F. De Carvalho, Y. Lechevallier, et A. Da Silva (2006a). Comparaison de dissimilarités pour l'analyse de l'usage d'un site web. *EGC 2006, RNTI-E-6 II*, 409–414.

Rossi, F., F. De Carvalho, Y. Lechevallier, et A. Da Silva (2006b). Dissimilarities for web usage mining. *IFCS 2006*, 39–46.

Spiliopoulou, M. (1999). Data mining for the web. *Workshop on Machine Learning in User Modelling of the ACAI99*, 588–589.

Tanasa, D. et B. Trousse (2004). Advanced data preprocessing for intersites web usage mining. *IEEE Intelligent Systems 19*(2), 59–65.

van Rijsbergen, C. J. (1979). *Information Retrieval* (second ed.). London: Butterworths.

Summary

Web usage analysis is very important for Web sites' operators as it provides some understanding of behaviour of their visitors. The way in which a site is visited can indeed evolve due to modifications of the structure and/or the contents of the site, or due to changes in the behaviour of certain user groups. Thus, the models associated with these behaviours must be updated continuously in order to reflect the current behaviour of the users. A solution to this problem, proposed in this article, is to update these models using the summaries obtained by an evolutionary approach of the classification method. For that, we carry out a split of the time into more significant time sub-periods. We compare the results obtained with this method to those produced by a global analysis.

Répartition optimisée des pas d'échantillonnage Application aux courbes de charge de consommation électrique

Raja Chiky* et Georges Hébrail*,**

* GET-ENST Paris
Laboratoire LTCI - UMR 5141 CNRS - Département Informatique et Réseaux
46 rue Barrault, 75634 Paris Cedex 13
Email: nom.prenom@enst.fr
** EDF R&D - Département ICAME
1, Avenue du Général de Gaulle, 92140 Clamart
Email: georges.hebrail@edf.fr

Résumé. Avec le développement de compteurs communicants, les consommations d'énergie électrique pourront à terme être télérelevées par les fournisseurs d'électricité à des pas de temps pouvant aller jusqu'à la seconde. Ceci générera des informations en continu, à un rythme rapide et en quantité importante. Si le système central collectant ces consommations n'est pas assez rapide pour traiter toutes les données des compteurs, plusieurs solutions peuvent être envisagées. Il est possible d'utiliser des buffers, et en cas de débordement de ces buffers, il faut ignorer certaines données, c'est le cas de l'échantillonnage. Nous présentons un algorithme d'échantillonnage en ligne basé sur l'apprentissage. Nous avons testé l'efficacité de notre algorithme sur un jeu de données réel de consommations électriques.

1 Introduction

Les courbes de charges sont utilisées pour étudier la consommation en électricité d'un client dans le temps. Il s'agit de l'évolution de la consommation d'énergie entre deux instants au cours du temps. La télérelève des courbes de charges se fait de façon continue à un pas de temps très fin pouvant aller jusqu'à la seconde. Ceci peut générer une quantité volumineuse de données, et il devient coûteux de stocker et traiter toutes ces données tant elles sont générées à un rythme rapide. Nous qualifions les courbes de charge par « flux de données », ce sont des séquences infinies de données arrivant sous forme de flux continus et ordonnés. Ces courbes de charges proviennent de compteurs électriques installés chez les clients et envoient leurs données simultanément à un système central. Celui-ci impose une limite de bande passante à ne pas dépasser et les données relevées à un pas de temps ne doivent pas excéder cette limite.
Le coût élevé de la collecte de données et la contrainte de la bande passante nous incitent à sélectionner un ensemble minimal de données échantillonnées tout en s'assurant de la représentativité de ces données pour s'approcher de la courbe de charge initiale. Le sur-échantillonnage est très couteux en temps et espace de stockage. D'autre part, le sous-échantillonnage mène à

une évaluation imprécise de la courbe de charge.
L'échantillonnage en ligne a été largement étudié dans le cadre des flux de données. Toutefois, la plupart des travaux se concentrent sur l'échantillonnage d'un flux de données individuelles. Dans notre cas, des centaines ou milliers de courbes de charge peuvent être mesurées simultanément et dépendent l'une de l'autre et nous ne pouvons décider d'une donnée à sélectionner dans une courbe de charge sans prendre en compte les données envoyées par les autres compteurs. Les méthodes traditionnelles d'échantillonnage des flux de données demeurent inefficaces dans notre cas.
Nous présentons dans ce papier, un algorithme d'échantillonnage en ligne basé sur l'apprentissage. Cet algorithme permet de déterminer les pas d'échantillonnage à l'instant t en prenant en compte des données à l'instant t-1. Nous avons démontré l'efficacité de notre algorithme en l'appliquant à des données de consommation électrique relevées à un pas de 10 minutes.
Ce papier est organisé comme suit : la section 2 présente un état de l'art sur l'échantillonnage dans les flux de données. La section 3 présente la formulation du problème, sa méthode de résolution est exposée à la section 4. Enfin, nous concluons à la section 5 en donnant nos perspectives de recherche.

2 Etat de l'art

L'échantillonnage dans les flux de données s'appuie sur les techniques d'échantillonnage traditionnelles, mais requiert des innovations significatives pour parer au problème de la longueur infinie des flux. En effet, des techniques de fenêtrage sont utilisées pour s'adapter à la nature illimitée des données : une fenêtre définit un intervalle temporel exprimé soit en terme de durée (par exemple les 5 dernières minutes), ou soit sous forme logique exprimé en nombre de tuples (par exemple les 20 derniers éléments). Ces fenêtres peuvent être délimitées par des bornes fixes ou glissantes dans le temps.
Quand on applique une méthode d'échantillonnage sur les flux de données, se pose le problème de l'expiration des éléments dans une fenêtre glissante. En effet, dans ces types de fenêtres les éléments ne faisant plus partie de la fenêtre courante deviennent invalides, et s'ils appartiennent à l'échantillon, il faut les remplacer. Des algorithmes permettant de tenir à jour un échantillon sur une fenêtre tout en préservant sa représentativité sont donc nécessaires. Plusieurs techniques ont été développées pour traiter le cas des fenêtres glissantes (logiques et temporelles). Un algorithme classique en ligne proposé par Vitter en 1985 est largement utilisé dans le cadre des flux de données : échantillonnage réservoir (Vitter, 1985). Il permet d'obtenir un échantillon uniforme aléatoire avec une taille fixe et ne nécessite pas de connaître la taille du flux de données. L'échantillonnage réservoir est utile dans le cas d'insertion ou de mises à jour mais trouve ses limites à l'expiration des données dans une fenêtre glissante. Une autre approche qualifée de 'simple' a été proposée dans (Babcock et al., 2002).Son principe est le suivant : on maintient un échantillonnage réservoir pour les premiers éléments du flux (première fenêtre). Et quand un élément expire, on le remplace par le nouvel élément qui arrive. Cet algorithme maintient un échantillon aléatoire uniforme pour la première fenêtre et ne requiert pas beaucoup de mémoire, mais a l'inconvénient d'être hautement périodique. Pour remédier à ce défaut, une technique a été proposée : échantillonnage avec réserve. Son principe est le suivant : quand un élément arrive on l'ajoute avec une probabilité donnée à un échantillon réserve ('backing sample') et on génère ensuite un échantillon aléatoire à partir de l'échantillon réserve.

Quand un élément expire, on le supprime de la réserve.
Une autre méthode adaptée au flux de données est le tirage de Bernouilli (Brown et al., 2006). Cette méthode a le mérite d'être simple à mettre en oeuvre mais a un inconvénient majeur qui est la variabilité incontrôlable de la taille de l'échantillon. Son principe consiste à effectuer une 'loterie' sur chaque individu de la fenêtre indépendamment d'un individu à l'autre. Ainsi, pour une fenêtre de taille N où les probabilités d'inclusion individuelles p_i sont connues pour tout i, on génère n réels aléatoires u_i entre 0 et 1 et on retient l'individu i si $u_i < p_i$.
En plus des algorithmes classiques tels que l'échantillonnage aléatoire simple ou l'échantillonnage stratifié, plusieurs autres algorithmes ont été développés pour les appliquer aux fenêtres logiques (tel que l'échantillonnage en chaîne (Babcock et al., 2002)) aux fenêtres temporelles (tel que l'échantillonnage par priorité (Babcock et al., 2002)) ou pour des conditions particulières d'utilisation (tel que le réservoir avec fréquences exactes (Gibbons et al., 1998)).

3 Formulation du problème

Soit *n* le nombre de courbes de charges (de compteurs). Chaque courbe est constituée de p points à la période $t-1$. Les courbes doivent être échantillonnées à un pas j inférieur à m, m permettant de définir une borne inférieure du nombre de données prélevées sur la période t. j correspond au 'saut' à effectuer entre deux points sélectionnés, par exemple $j = 2$ signifie que nous sélectionnons un point sur deux de la courbe de charge. Soit s le nombre de points communicables au système central sur une période donnée ($s < n * p$), c'est à dire le nombre de points que le système central peut accepter pendant la période.
Nous cherchons à déterminer un planning de collecte de données à effectuer pendant une période t (journée par exemple) au niveau global. Le problème consiste à trouver des pas d'échantillonnage pour chaque compteur en respectant les données précédentes (Nombre de données communicables et pas d'échantillonnage minimum). Les pas sont calculés sur les courbes de charge de la période $t-1$. En effet, nous estimons que les données des courbes de charge sont fortement corrélées pendant deux périodes successives.
Les pas d'échantillonnage doivent permettre de représenter la courbe de charge initiale le plus finement possible. Le problème d'échantillonnage consiste donc à minimiser la somme des erreurs quadratiques entre la courbe de charge d'origine à la période $t-1$, et la courbe échantillonnée en prenant en compte les contraintes citées ci-dessus.
Plus précisément, nous recherchons une approximation $\hat{C}_i$ de C_i (courbe de charge du compteur i), telle que :

$$ArgMin(\sum_{i=1}^{n} SSE(C_i, \hat{C}_i))$$

où n est le nombre de courbes de charges et SSE est la somme des erreurs quadratiques entre C_i et le modèle $\hat{C}_i$. SSE se calcule de la façon suivante :
$SSE(C, \hat{C}) = \sum_{i=1}^{p}(c_i - \hat{c}_i)^2$, *p* est le nombre des points de la courbe C.
Cette minimisation des SSE doit se faire avec les contraintes :

- $\sum_{i=1}^{n} \left\lfloor \frac{p}{j_i} \right\rfloor \leq s$, *s* est la limite des données à communiquer et j_i les pas d'échantillonnage.
- $\forall i \in 1, 2, ..., n\ j_i \leq m$.

Une fois les pas d'échantillonnage déterminés, se pose le problème d'envoi de données au premier pas de temps. En effet, tel que le problème est formulé, toutes les courbes de charges envoient simultanément une valeur de consommation électrique au premier pas de temps, et nous dépassons ainsi la limite de la bande passante. Nous ne nous intéressons pas à ce problème dans le cadre de ce papier, mais nous pouvons facilement le détourner, par exemple en effectuant au premier pas de temps, un échantillonnage aléatoire des courbes de charges qui devront envoyer en même temps les données de consommation.

4 Méthode de résolution

La méthode la plus immédiate pour déterminer les pas d'échantillonnage est de partager la bande passante sur les différentes courbes de charges, les pas d'échantillonnage étant les mêmes pour toutes les courbes. Cette solution respecte bien les contraintes posées mais ne minimise pas la somme des erreurs quadratiques entre les courbes initiales et les courbes échantillonnées. De plus, les courbes présentant des fluctuations différentes, les pas d'échantillonnages « fixes »pourraient sur-échantillonner une courbe ou au contraire la sous-échantillonner. Pour résoudre le problème de minimisation des SSE, nous l'avons modélisé sous forme d'un problème d'optimisation linéaire. Voici comment nous avons procédé :
Nous disposons de *n* courbes, chacune contenant *p* points (consommation à chaque pas de temps). Chaque courbe doit être échantillonnée à un pas inférieur ou égal à *m* (m étant inférieur à p). Nous avons donc la possibilité de choisir un pas allant de 1 (nous récupérons tous les points de la courbe de charge) à *m* (les points échantillonnés sont distancés par *m*). Nous calculons une matrice W_{n*m} de *n* lignes et *m* colonnes. Un élément w_{ij} de la matrice correspond à la somme des erreurs quadratiques obtenue si on appliquait un pas d'échantillonnage *j* à la courbe d'indice *i*.
Par conséquent, le problème à résoudre est :

$$\text{Minimiser } \sum_{i=1}^{n} \sum_{j=1}^{m} (W_{ij} \times X_{ij})$$

sous les contraintes :

$$\begin{cases} X_{ij} = 0 \text{ ou } 1 & \\ \sum_{j=1}^{m} X_{ij} = 1 & \text{i de 1 à n} \\ \sum_{i=1}^{n} \sum_{j=1}^{m} (\left\lfloor \frac{p}{j} \right\rfloor \times X_{ij}) \leq s & \text{i de 1 à n} \end{cases}$$

Il s'agit d'un problème d'affectation des pas d'échantillonnage sur les différentes courbes en respectant les contraintes ci-dessus. Une variable X_{ij} à 1 signifie que nous affectons le pas d'échantillonnage j à la courbe i. La deuxième contrainte $\sum_{j=1}^{m} X_{ij} = 1$ traduit une seule valeur de j par courbe (un seul pas d'échantillonnage par courbe). Enfin, la troisième contrainte signifie que la somme des pas d'échantillonnage ne doit pas dépasser le seuil imposé (le nombre de données à communiquer au système central).
Pour résoudre ce problème, nous avons utilisé un programme LP_Solve qui permet de résoudre des problémes linéaires à variables entières et/ou réelles. Pour les variables entières comme dans notre cas, ce programme utilise la méthode Branch And Bound (Séparation-Evaluation

en français), couplé à la méthode du simplex connu pour les résolutions des problèmes linéaires à variables réelles (le lecteur peut se référer à (Gondran et al., 1979; lpsolve) pour des informations sur ces méthodes).

5 Expérimentations

Nos expérimentations ont été faites sur un jeu de données composé de consommations électriques relevées à pas de 10 minutes pendant une journée (144 relèves par compteur électrique). Il s'agit d'un fichier de 1058 compteurs électriques appartenant à des hôtels. Les courbes de charge ont été normalisées.

5.1 Matrice des erreurs

La somme des erreurs quadratiques dépend de la façon dont nous avons construit la courbe échantillonnée. Nous avons choisi deux méthodes et les avons comparées entre elles. Les résultats sont exposées à la section 5.2.
Soient une courbe de charge *C* contenant *p* points $C = \{c_1, c_2, ..., c_p\}$ et $\hat{C}$ sa courbe échantillonnée à un pas j $\hat{C} = \{\hat{c}_1, \hat{c}_2, ..., \hat{c}_p\}$.
La première méthode construit la courbe échantillonnée sous forme d'escalier, les points compris entre deux données sélectionnées successives c_a et c_b prennent la valeur de la première donnée c_a. C'est à dire que la relation entre C et $\hat{C}$ est donnée par :

$$\hat{c}_i = \begin{cases} c_i & \text{si } i \; modulo \; j = 1 \\ c_{i-1} & \text{sinon} \end{cases}$$

La deuxième méthode construit la courbe échantillonnée par interpolation linéaire, les valeurs des points compris entre deux données sélectionnées successives c_a et c_b sont calculées par interpolation linéaire en utilisant les valeurs de c_a et c_b :

$$\hat{c}_i = \begin{cases} c_i & \text{si } i \; modulo \; j = 1 \\ f(i) & \text{sinon} \end{cases}$$

Soient c_a et c_b les deux points échantillonnés de C tel que $a < i < b$. La fonction f est la droite d'interpolation linéaire entre les deux points c_a et c_b, et a pour équation :
$f(i) = \frac{i-b}{a-b} c_a - \frac{i-a}{a-b} c_b$
La figure 1 montre un exemple de courbe de charge, et ses courbes échantillonnées en escalier et par interpolation linéaire.

5.2 Résultats

Nous avons considéré *m=20* ce qui signifie que pour chaque compteur, au moins un index de consommation sur 20 doit être sélectionné. La courbe échantillonnée sera donc constituée d'au moins 7 valeurs puisque nous rappelons qu'une courbe de charge est constituée de 144 points. Nous avons testé plusieurs valeurs de bande passante (seuil s) et nous avons dressé un

tableau récapitulatif des résultats obtenus (tableau 1). Chaque colonne du tableau correspond à une valeur du seuil s. Nous avons calculé la somme des erreurs quadratiques pour chacune des méthodes ('escalier', 'interpolation linéaire', 'fixe'), la moyenne des pas d'échantillonnage, l'écart-type pour mesurer la dispersion des pas d'échantillonnage autour de la moyenne, et enfin la moyenne des différences des pas d'échantillonnage entre la méthode 'escalier' et la méthode 'interpolation linéaire'.

	80610	53645	32210	23014	16122	10744
SSE opt esc	30.11	97.19	303.55	619.73	1385.24	3898.74
SSE opt IL	12.02	43.51	143.80	303.66	719.66	2044.54
SSE fixe esc	2801.03 (pas=2)	4386.46 (pas=3)	8250.06 (pas=5)	9482.21 (pas=7)	13602.96 (pas=10)	16466.54 (pas=15)
SSE fixe IL	1970.25 (pas=2)	2856.15 (pas=3)	4311.07 (pas=5)	5596.45 (pas=7)	7188.64 (pas=10)	9283.58 (pas=15)
moyenne pas esc	1.89	2.84	4.73	6.62	9.45	14.18
ecart-type pas esc	3.78	5.28	7	8.08	8.63	7.62
moyenne pas IL	1.89	2.84	4.73	6.62	9.45	14.18
ecart-type pas IL	3.86	5.42	7.27	8.37	9.12	8.35
moyenne diff. pas esc-IL	0.52	1.25	1.66	1.49	2.3	2.58

TAB. 1 – *tableau récapitulatif. SSE : Somme de l'écart quadratique. opt : pas d'échantillonnage obtenus par optimisation, et fixe : même pas d'échantillonnage pour les CDCs. esc : courbe échantillonnée en escalier et IL : par Interpolation Linéaire. La dernière ligne correspond à la moyenne des différences entre les pas d'échantillonnage en escalier et avec IL.*

Les résultats du tableau 1 font apparaître que l'optimisation permet de diminuer considérablement la somme des erreurs quadratiques que ce soit avec la méthode en 'escalier' ou avec l'interpolation linéaire par rapport à l'échantillonnage au pas 'fixe'. Par exemple, en utilisant un seuil de 80610, l'optimisation avec 'Interpolation Linéaire' nous permet de minimiser l'erreur quadratique de 99% par rapport à l'échantillonnage 'fixe'.

Quoique les moyennes des pas d'échantillonnage par optimisation se rapprochent des pas 'fixes', nous observons des valeurs élevées des écarts-types. Les pas d'échantillonnage sont

largement écartés de la moyenne. La figure 2 est un exemple de distribution des pas d'échantillonnage ('escalier' et 'interpolation linéaire') pour un seuil de 10000. Notons que les pas d'échantillonnage par optimisation sont répartis sur les différentes valeurs des pas avec une concentration au niveau des limites (*j=1* et *j=20*), alors qu'en échantillonnage 'fixe' le pas d'échantillonnage est *j=10*. Ce fait permet de confirmer l'importance de sélectionner les pas d'échantillonnage par optimisation.

Il est aussi intéressant de remarquer qu'il n'y a pas de grandes différences en moyenne entre les pas d'échantillonnage affectés par la méthode escalier et la méthode utilisant l'interpolation linéaire. Nous pouvons donc utiliser indifféremment l'une ou l'autre dans nos prochaines expérimentations.
Cependant, la figure 3.a présente l'évolution de la somme de l'erreur quadratique en fonction du pas d'échantillonnage calculé à partir de la courbe de charge de la figure 3.b. Nous observons une fluctuation du SSE en fonction du pas d'échantillonnage. Cette fluctuation est plus apparente dans le cas de l'échantillonnage 'escalier' que dans le cas de l'interpolation linéaire. Dans l'exemple de la figure 3, le programme d'optimisation va ainsi choisir le pas d'échantillonnage 7 au lieu de 4,5 ou 6 pour minimiser l'erreur quadratique. Il est donc préférable d'utiliser la méthode où les variations sont moindres, il s'agit ici de l'interpolation linéaire.

5.3 Echantillonnage dans le temps

Afin d'appliquer le programme de collecte de données sur des flux de données arrivant en continu, il faut mettre à jour la matrice des erreurs d'échantillonnage à l'expiration de la fenêtre temporelle. Chaque compteur envoie les informations nécessaires et suffisantes à la mise à jour de la matrice. Il faudra ainsi prendre en compte l'espace mémoire occupé par ces informations dans le calcul des prochains pas d'échantillonnage.
Pour mettre à jour la matrice des erreurs, chaque compteur devrait envoyer m SSEs (nous rappelons que m correspond à la limite inférieure du pas d'échantillonnage). Cependant, il n'est pas nécessaire d'envoyer la totalité des m SSEs. En effet, une courbe de charge échantillonnée à un pas j, envoie $m - M_j$ SSEs, M_j étant le nombre de multiples de j qui sont inférieurs à m. Etant donné que le système central dispose des données de consommation échantillonnées, il peut donc calculer à partir de ces données et des SSEs envoyées par le compteur les SSEs manquant à la mise à jour complète de la matrice. A titre d'exemple, une courbe de charge échantillonnée à $j = 2$ et $m = 20$ envoie à la fin de la période $m - M_2 = 20 - 10 = 10$ SSEs, tel que $M_2 = card(\{2, 4, 6, ...20\}) = 10$. Dans nos expérimentations, pour un seuil de $s = 16122$ et $m = 20$, chaque compteur envoie en moyenne 9 SSEs au lieu de 20.

6 Conclusion et perspectives

La mise en oeuvre de l'échantillonnage appliqué à des courbes de charges a permis de montrer que l'affectation des pas d'échantillonnage par optimisation linéaire permet de réduire significativement les erreurs d'échantillonnage par rapport à un échantillonnage à pas 'fixe'. Notre problématique s'insère dans le cadre d'échantillonnage des flux de données (données de consommation électrique en ligne) en spécifiant des fenêtres temporelles glissantes (courbe de charge d'une période t).
Tel que le problème a été formulé à la section 4, les pas d'échantillonnage obtenus grâce aux

courbes de charge de la période t, devraient être appliqués pour échantillonner les courbes de charge à la période t+1. Or, nous avons effectué l'étape de l'optimisation et l'étape d'échantillonnage aux mêmes périodes lors de nos expérimentations. Nous y avons été contraints à cause de la taille du jeu de données dont nous disposons. En perspective, il serait intéressant d'augmenter la taille du jeu de données en prenant par exemple une courbe de charge relevée à pas de temps de l'ordre d'une seconde et tester plusieurs tailles de fenêtres temporelles (journée, semaine, ...).
Deux autres axes de recherche sont l'ajout de contraintes soulignant les corrélations entre des courbes de charges données, ou encore l'application des méthodes d'échantillonnage adaptatif au niveau de chaque courbe. Ainsi, une courbe de charge peut être échantillonnée à un pas variable dans le temps selon ses changements.

Références

Babcock B., Datar M., Motwani R. (2002). Sampling From a Moving Window Over Streaming Data, 2002 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2002).

P. G. Brown et P. J. Haas (2006). Infrastructure and algorithms for warehousing of sample data. Proc. 22nd Intl. Conf. Data Engrg., 2006.

Phillip B. Gibbons et Yossi Matias (1998). New sampling-based summary statistics for improving approximate query answers. In Proc. ACM SIGMOD Conf., Seattle, WA, USA, June 1998.

M.Gondran et M.minoux. (1979). Graphes et algorithmes. Eyrolles, Paris 1979.

http ://lpsolve.sourceforge.net/

Vitter, J. S. (1985). Random sampling with a reservoir. ACM rans. Math. Softw. 11, 1, 35-57

Summary

With the development of AMM (Automatic Metering management), it is now possible for electric power suppliers to acquire from customers their electric power consumption as frequently as every second. This will generate data arriving in multiple, continuous, rapid, and time-varying streams. Several solutions can be considered if the central system collecting electric consumption data is not fast enough to process all the incoming data. It is possible to use buffers, and reject some data in the case of overflow using sampling techniques. We present a learning based online sampling algorithm. We tested the effectiveness of our algorithm on a real electric consumption data file.

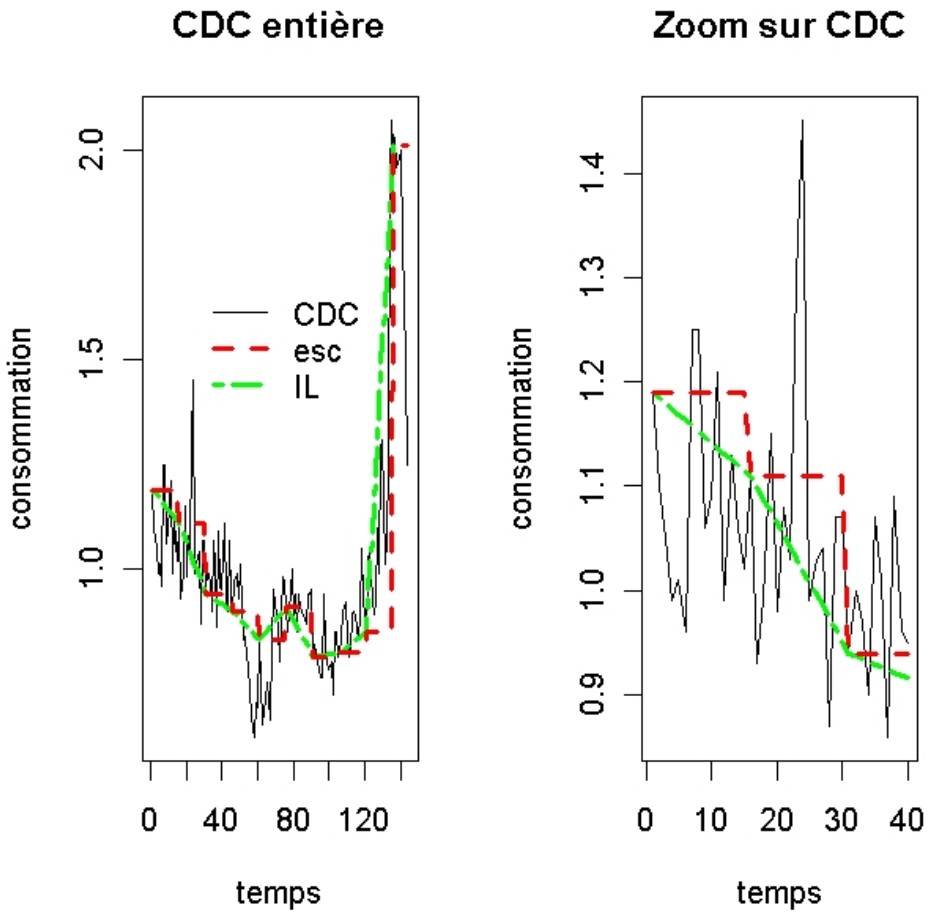

FIG. 1 – *Une Courbe de charge et ses courbes échantillonnées à pas=15 en Escalier (esc) et Interpolation linéaire (IL). La figure de gauche correspond aux 40 premiers points de la CDC.*

Répartition optimisée des pas d'échantillonnage

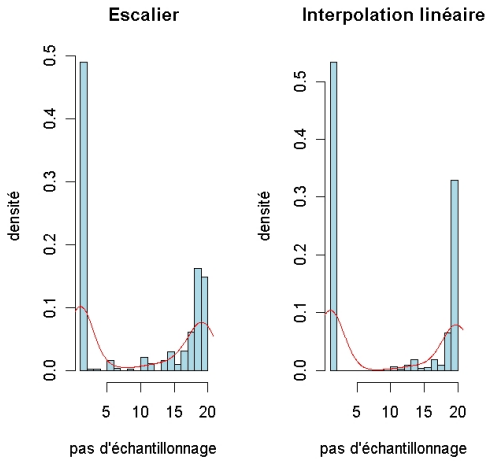

FIG. 2 – *Densité de distribution des pas d'échantillonnage*

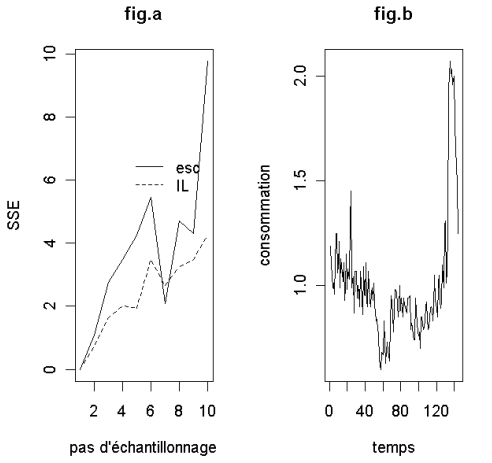

FIG. 3 – *fig. a : SSE en fonction des pas d'échantillonnage. fig. b : CDC utilisé pour le calcul des SSE*

Data Stream, Stream-Mining et prévision de la consommation électrique à EDF : Réflexions préliminaires et pistes de recherche

Alain Dessertaine

EDF R&D OSIRIS
1 avenue du Général De Gaulle
92141 Clamart Cedex
alain.dessertaine@edf.fr

Résumé. Dans cet article, nous mettons en évidence l'intérêt d'utiliser les données issues des futurs 32 millions de compteurs communicants qui seront installés chez l'ensemble des consommateurs de France d'ici 2013 afin de construire des prévisions Court-termes et Moyens-termes de la consommation électrique de EDF (global, ou par portefeuilles) dans un environnement de flux de données. Nous développons nos premières réflexions et évoquons quelques pistes de recherche. Ces pistes devraient nous permettre d'aborder des modélisations et prévisions par agrégation / désagrégation de courbes, ainsi que des modélisations et prévisions sur données hilbertiennes. Nous mettons en regard ces idées avec les approches de type Stream-Mining.

1 Contexte

1.1 Quelques mots sur l'utilisation potentielle des flux de données à EDF

Le volume de données traitées et analysées par EDF devient de plus en plus important. La mise en place de systèmes de mesure de plus en plus performant augmentera ce volume de manière conséquente, avec une volonté d'avoir un éclairage sur ces données et sur les concepts qu'elles décrivent au fil de l'eau pour une meilleure réactivité de certaines prises de décision. Ainsi, par exemple, une montée en compétence sur l'utilisation et la modélisation de flux de données structurées devrait permettre de calculer et d'analyser des indicateurs de surveillance et de performances des centrales de production, dans un environnement de flux de données.

De plus, l'installation prévue d'ici 2013 de plus de 32 millions de compteurs communicants sur l'ensemble des consommateurs d'Electricité de France devrait permettre d'analyser au mieux les habitudes de consommation de la clientèle d'EDF, mais aussi d'analyser « en

direct » la consommation à des niveaux plus ou moins agrégés et de la prévoir afin d'adapter l'appareil de production et les achats potentiels sur les marchés de l'électricité. En effet, ces compteurs serviront entre autres de capteurs afin de mesurer l'ensemble de toutes les courbes de charges de chaque client à des granularités temporelles très fines (allant jusqu'à la minute, voire la seconde), alors que, aujourd'hui, seuls quelques dizaines de milliers de clients[1] possèdent des compteurs dits « télé-relevables » permettant ainsi de récupérer (avec un certain délai) des courbes avec des mesures toutes les demi-heures, voire toutes les dix minutes. C'est l'utilisation potentielle de ces futures données qui nous amène à réfléchir et à construire des pistes de travail et de recherche que nous allons succinctement présenter dans cet article.

1.2 La prévision de la consommation électrique : Pourquoi et comment?

La prévision de la consommation électrique est de très haute importance pour EDF. La prévision de long terme permet de gérer les futurs investissements pour faire évoluer de manière adéquate le parc de production, tandis que les prévisions de court terme (de 1 heure à une dizaine de jours) et de moyen terme (plus de 10 jours à quelques mois) permettent d'adapter le pilotage des outils de production et d'achat sur les marchés financiers de l'électricité à la consommation de son portefeuille de clients. Dans les deux derniers cas, nous prévoyons des consommations globales ou par portefeuille à des pas horaires ou demi-horaires.

Vu l'évolution du contexte concurrentiel du marché de l'électricité, le métier du prévisionniste dans le secteur de l'énergie a fortement évolué et la prévision de consommation est devenue multiforme compte tenu de nouveaux besoins : d'une prévision nationale au pas demi horaire avec une connaissance parfaite du passé et avec des variables explicatives connues (telles les variables de températures et de nébulosité), nous devons maintenant effectuer des prévisions individuelles ou de portefeuille avec des données différentes, évolutives et une augmentation de la nécessité de précision compte tenu des enjeux financiers (gestion de parc et sourcing sur des marchés financiers, proposition d'offres individualisées, mécanisme d'ajustement et de règlement des écarts etc…) ;

Aussi, des efforts sont en cours afin d'adapter les modèles existants (modèles généralement non linéaires prenant en compte les effets de la température, de la nébulosité, mais aussi de données calendaires en ajoutant une modélisation de la tendance et des phénomènes saisonniers et périodiques) afin de prévoir des consommations les plus individuelles possibles, d'agréger ces consommations au mieux, mais aussi de prévoir au mieux des consommations agrégées sur des portefeuilles volatils et de maintenir, voire améliorer des prévisions sur les signaux globaux.

L'utilisation des données provenant des quelques 32 millions de compteurs communicants devrait permettre de construire de nouveaux outils d'aide à la décision et de prévision, en adaptant ou en utilisant les modèles et outils issus des approches de traitements des flux de données. La mise en place sur le marché de système permettant la gestion et la modélisa-

[1] Ce sont les clients les plus importants, ainsi que plusieurs échantillons de clients panélisés à la fois pour analyser leurs usages, mais aussi pour construire des profils institutionnels sur lesquels sont basés, actuellement, les calculs de « clés de répartition » pour dispatcher la consommation globale constatée sur l'ensemble des distributeurs d'électricité du marché français.

tion d'une telle structure de données devrait devenir effective dans les années à venir. Aussi, l'enjeu pour EDF sera donc différent pour les composantes décisionnelles déjà développées et celles qui ne le sont pas encore. Pour les composantes décisionnelles à développer dans un futur proche, les logiciels commerciaux de gestion des flux de données ne seront pas encore disponibles. L'enjeu consiste alors à bien connaître les principes et algorithmes – aujourd'hui présents dans les prototypes de recherche – afin d'être précurseurs dans un traitement des informations au fil de l'eau lorsque cela est possible.

Ce document doit présenter les premières réflexions ainsi que les axes de recherche que la Direction de la Recherche et de Développement d'EDF a décidé de mettre en place pour pouvoir être en mesure d'exploiter au mieux ces données.

2 Premières réflexions et quelques pistes de recherche

2.1 Quelques remarques sur les données de base

La grande richesse de ces futures données sera liée à leur caractère spatio-temporel, d'une part, et à leur caractère temporel « quasi-continu » d'autre part. Ce dernier caractère donnera forcément un éclairage des plus importants sur le phénomène étudié ; en effet, nous pourrons vérifier si l'échantillonnage temporel utilisé aujourd'hui pour faire des prévisions (au pas horaire ou demi-horaire) est le plus adapté pour bien analyser la consommation électrique d'un portefeuille[2] .

De plus, il est prévu de pouvoir capter, pour certains clients l'ayant préalablement accepté (de manière contractuelle ou autre…) des données par usages (comme le chauffage, le chauffe eau, cuisine, etc…) mais aussi des données de températures à l'intérieur des logements (voir à l'extérieur…).

Les besoins de facturation et de règlement des écarts[3] nécessiteront des récupérations de données, dont les règles n'ont pas été au jour d'aujourd'hui déterminées. Pourrons-nous utiliser des données exhaustives d'un côté (données quotidiennes, voire horaire pour l'ensemble des clients), et les résultats d'échantillons « spatio-temporels » pour reconstruire des historiques à des granularités plus fines (minute, seconde …) ?

Les incertitudes concernant les données potentiellement exploitables devront guider nos choix, au fur et à mesure, vers certaines voies de recherche plutôt que vers d'autres.

Par contre, il est clair que nous pouvons prendre en compte, au jour d'aujourd'hui, les caractères spatio-temporel et quasi-continu de nos données pour élaborer nos premières réflexions et travaux de recherche.

---

[2] Par exemple, nous observons sur la courbe globale France une pointe de consommation en fin d'après-midi, correspondant à l'addition de diverses phénomènes comme l'allumage de l'éclairage, l'utilisation des plaques et fours électriques pour effectuer la cuisine du soir, l'utilisation de certains chauffe-eau etc… La prévision de cette pointe est primordiale, particulièrement en hiver. Or, elle se « déplace » constamment dans le temps au fur et à mesure de l'année (et des années). Ceci est dû au fait, entre autres, des heures variables du coucher du soleil. Ce phénomène serait, sans nul doute, plus facilement observable et modélisable en utilisant des données « quasi-continues ».

[3] Terme donné au « dispatching » de la consommation sur l'ensemble des distributeurs d'électricité, évoqué ci-dessus.

2.2 Quelques remarques sur les flux et résumés de données

La grande volumétrie des données que nous venons d'aborder impliquera des contraintes très fortes en terme de transmission et de stockage de la donnée. Des réflexions seront donc nécessaires pour définir nos besoins en terme de données à exploiter pour élaborer des outils de prévision pertinents.

Nous pouvons cependant donner quelques remarques et préconisations préliminaires :

- Nous aurons besoin de constituer des historiques relativement longs afin de capter les phénomènes temporels tels les tendances, les saisonnalités et autres phénomènes périodiques de nos données (les granularités temporelles pourront être d'autant plus larges que les données stockées seront éloignées dans le passé)
- Nous aurons besoin de travailler à des mailles suffisamment agrégées afin d'utiliser certaines propriétés intéressantes liées au foisonnement (nous donnerons une définition possible du foisonnement ci-après).
- Nous aurons besoin de relier les courbes captées à des données fiables permettant de qualifier nos données :
    - données géographiques (voire socio-économiques)
    - données contractuelles (voire information sur les usages)
    - données météorologiques : mesures, voire données prévisionnelles

Des travaux sont en cours afin de proposer des stratégies d'échantillonnage spatio-temporelles des flux utilisés[4] ; nous allons nous baser sur ces premiers travaux pour prendre en compte certaines contraintes préalables à nos futurs traitements.

Aussi, nous pouvons au jour d'aujourd'hui dire que nous ne pourrons pas conserver (ni récupérer) l'ensemble des courbes de consommation au pas seconde ou au pas minute sur l'ensemble de nos clients. Nous ferons comme hypothèse de travail que nous gèrerons un panel avec un grand nombre de clients, et que nous récupèrerons des résumés spécifiques de leur(s) courbe(s) de consommation, en fonction de la richesse même de leur propre process de consommation. Ainsi :

- nous pourrons récupérer des courbes avec des granularités temporelles différentes, mais constantes tout au long de certaines périodes de chaque courbe (la fenêtre temporelle pourra être, par exemple, quotidienne) ; ces granularités choisies de manière à optimiser les erreurs quadratiques globales (Chiky, 2007).
- si la granularité peut être préétablie au niveau du compteur, nous pourrons peut-être récupérer des courbes à granularités variables dans le temps pour chaque courbe de l'échantillon afin de bien capter certaines périodes d'augmentation, de diminution ou de fortes variabilités de la consommation individuelle.
- nous pourrons aussi récupérer des résumés à base de décompositions fonctionnelles (sur des bases d'ondelettes, par exemple). Les compressions pourront elles-mêmes dépendre de la richesse de chaque signal importer et/ou de chaque instant étudié.

Il reste à voir comment nous pourrons exploiter ces restitutions de flux de données.

[4] Thèse de Raja Chiky (ENST) sous la direction de Georges Hebrail

2.3 Quelques idées préliminaires pour l'exploitation des flux de données pour la construction de modèles de prévision

Construire des prévisions individuelles afin d'agréger ces prédictions pour élaborer la prévision d'ensemble (en utilisant des pondérations adéquates) sera sans nulle doute inadapté. En effet, certaines études sur des courbes individuelles nous ont prouvé que les consommations individuelles étaient très difficiles à modéliser, dû à une grande variabilité de celles-ci, et à un caractère fortement aléatoire et erratique. Comme nous l'avons remarqué plus haut, il nous faudrait pouvoir regrouper les courbes de manière intelligente afin d'utiliser les propriétés de foisonnement pour que la somme (ou l'estimation de la somme) des courbes de chaque groupe soient bien prévisibles.

Cette piste peut être éventuellement assimilée à une problématique de classification supervisée de courbes par arbre, avec prise en compte à la fois de variables exogènes (dont les coordonnées spatiales et, si possible, la température intérieure et/ou extérieure), et, surtout, d'un indicateur (à définir) permettant d'élaguer l'arbre à un niveau permettant de respecter l'objectif d'amélioration de prédictibilité du signal global après empilement des prévisions au niveau de chaque classe construite ! La variable à expliquer ne serait pas une simple variable continue, mais une variable composée de fonctions. Des liens seront sans doute à construire avec les travaux présentés dans Ramsay et Silverman (2002) ainsi que Ramsay et Silverman (2005).

De plus, la mise à jour au fil de l'eau demanderait de réaliser de manière incrémentale cette classification supervisée.

2.3.1 Une formalisation possible du problème

- Notations **:**

Soit i, un individu sur lequel est mesurée une courbe de consommation d'électricité $C_i(t)$.

La consommation totale de la population d'étude peut s'écrire (nous appellerons la courbe correspondante une synchrone):

$$C(t) = \sum_{i=1}^{N} C_i(t) \qquad (1)$$

avec N nombre total d'individus de notre population d'étude. Nous parlerons aussi du nombre total de courbes disponibles aux instants d'étude. Remarquons, ici, que les courbes C_i ne seront pas forcément mesurées avec le même pas de temps, ou avec la même finesse. D'où l'obligation d'établir des lissages fonctionnels adaptés pour construire cette somme.

La population d'étude peut s'éclater, sur une période T donnée, en G_T sous-groupes g de N_g clients. Nous pouvons définir les consommations par groupe (sous-synchrones) de la manière suivante :

$$C(t) = \sum_{g=1}^{G_T} C_g(t) \qquad (2)$$

avec :

$$C_g(t) = \sum_{i=1}^{N_g} C_i(t) \quad (3)$$

Remarquons d'ores et déjà que si nous ne travaillons qu'avec un échantillon de courbes de clients, nous pouvons calculer des estimations instantanées de ces consommations totales si nous connaissons un jeu de poids w_i : ces poids seront, dans un premier temps, les inverses de probabilités d'inclusions des individus à nos échantillons, directement issues du plan de sondage utilisé pour construire notre panel[5]. Auquel cas nous avons:

$$\hat{C}(t) = \sum_{i=1}^{n} w_i C_i(t) \quad (4)$$

$$\hat{C}_g(t) = \sum_{i=1}^{n_g} w_i C_i(t) \quad (5)$$

avec n et n_g respectivement les nombres de clients dans l'échantillon global, ou dans l'échantillon sur le groupe g. La formule (5) revient à effectuer l'estimation de la synchrone sur ce que l'on appelle, en théorie des sondages, le domaine défini par le groupe g. Bien entendu, des liens pourront être établis afin d'utiliser toutes les informations en notre possession afin d'estimer au mieux $\hat{C}_g(t)$, particulièrement dans le cas d'un petit domaine. Pour plus de détails, nous pourrons consulter Deville et Särndal (1992), Lundström et Särndal (2005) et Rao (2003).

Dans les deux cas, nous pouvons calculer, voire estimer les erreurs d'échantillonnage des estimateurs. Un plus à apporter ici sera de pouvoir réduire au mieux les erreurs d'échantillonnage en utilisant toutes les informations en notre possession soit en construisant des échantillons équilibrés, soit en effectuant des redressements afin de conserver au mieux dans nos échantillons et nos estimations la « représentativité » temporelle de notre sondage. Des premiers travaux ont été menés dans ce sens afin de prendre en compte des données de type courbe comme variables auxiliaires pour construire des re-pondérations par calage sur marge (voir Dessertaine, 2006), ou par échantillonnage équilibré (voir Dessertaine, 2007).

- Classification supervisée pour la prévision par agrégation / désagrégation de courbe

Nous partons de l'idée qu'il existe une partition de la clientèle en G_T groupes telle que nous pourrions modéliser chaque signal $C_g(t)$ de manière à ce que la somme des prévisions calculées sur la période T soit de manière générale plus pertinente et de meilleure qualité (au sens du RMSE, par exemple) que la prévision réalisée à partir de la modélisation adaptée sur le signal agrégé C(t).

Sur les signaux de consommations électriques, nous pouvons illustrer ce phénomène par la prise en compte spécifique de variables météorologiques locales, par exemple, pour modéliser les consommations des clients thermosensibles du même périmètre. D'autres variables plus spécifiques (socio-économiques, démographiques, contractuelles etc…) devront bien évidemment être pris en compte à ce niveau !

[5] A ce niveau, il faudra bien entendu tenir compte des erreurs de transmission des données générant des valeurs manquantes, voire de certaines valeurs remarquables ou aberrantes des informations transmises

Une autre manière de décrire ce principe serait de dire que pour un individu i appartenant à un groupe g, son signal de consommation pourrait se décomposer de manière additive en deux sous signaux aléatoires $S_g(t)$ et $\xi_i(t)$:

$$C_i(t) = \lambda_i S_g(t) + \xi_i(t) \tag{6}$$

nous avons, ainsi, un signal individuel composé comme la somme d'un signal commun à tous les individus du groupe g, à un coefficient multiplicatif près, et d'un sous-signal propre. A ce niveau, je propose que nous fassions l'hypothèse de l'indépendance sur le temps des signaux propres entre eux suivante :

$$Cov_t(\xi_i(t), \xi_j(t)) = 0 \quad \forall (i,j) \in g \tag{7}$$

avec :

$$Cov_i(\xi_i(t), \xi_j(t)) = \int_T \left(\xi_i(t) - \frac{1}{T}\int_0^T \xi_i(k).dk \right)\left(\xi_j(t) - \frac{1}{T}\int_0^T \xi_j(k).dk \right) \tag{8}$$

Dans ce cas, nous pouvons dire que :

$$C_g(t) = S_g(t)\sum_{i=1}^{N_g} \lambda_i + o(S_g(t)\sum_{i=1}^{N_g} \lambda_i) \tag{9}$$

Ceci peut être une « définition » du foisonnement entre courbes, généralisée à un regroupement particulier d'individus.
Dans le cas où nous ne travaillons pas avec l'exhaustivité des individus, nous pouvons dire que, en utilisant le jeu de poids w_i :

$$\hat{C}_g(t) = \hat{S}_g(t)\sum_{i=1}^{n_g} w_i\lambda_i + o(\hat{S}_g(t)\sum_{i=1}^{N_g} w_i\lambda_i) \tag{10}$$

Supposons que nous puissions construire des modèles « parfaitement adaptés » à chaque signal $S_g(t)$, une désagrégation de notre signal de départ pourrait se traduire par le fait que le prédicteur composite des prédicteurs construits sur ces G modèles indépendants soient à tout instant (ou en moyenne sur une période donnée) plus performant que le « meilleur » prédicteur construit sur la connaissance du signal global C(t) et des variables. Si nous mesurons la performance par une fonction Q (RMSE de prévision sur une période p donnée par exemple), nous avons alors :

$$\bar{Q}(\sum_{g=1}^{G_T} \hat{S}_g(t)\sum_{i=1}^{N_g} \lambda_i, t \in p) \le \bar{Q}(\hat{C}(t), t \in p) \tag{11}$$

Cette valeur dépend non seulement de la performance de chaque prédicteur, mais aussi de l'erreur « d'échantillonnage » effectuée sur le terme $S_g(t)\sum_{i=1}^{N_g} \lambda_i$, si celui-ci est estimé par la valeur $\hat{S}_g(t)\sum_{i=1}^{n_g} w_i\lambda_i$.

Quelques idées pour la classification :

Le problème peut consister à construire et/ou maintenir une partition en G_T groupes respectant les hypothèses ci-dessus, de manière à vérifier (11). Ainsi, nous pouvons soit construire une classification non supervisée ou supervisée des signaux individuels disponibles en G_T classes, de manière pertinente.
Aussi, le signal $S_g(t)$ pourrait être approché par le signal moyen de chaque signal individuel appartenant à la classe g, ce à chaque instant t. Il « suffirait », dans le cadre d'une partition hiérarchique par exemple, d'introduire un indicateur de coupure de l'arbre reflétant la « contrainte » (11).

Plusieurs questions se posent :
- Compression adaptée de chaque courbe ; des essais ont été réalisés en utilisant des débruitages et des compressions à base d'ondelettes donnant sur un jeu de 2300 courbes des résultats prometteurs. Ces essais ont utilisé des principes et des algorithmes décrits et proposés dans Misiti et al., (1998).
- Construction des $\square_i(t)$ vérifiant (7)
- Comment formaliser un indicateur de coupure permettant de respecter (11)

D'autres points devront sans doute être abordés à ce niveau (comme la prise en compte dans l'algorithme de classification du caractère prédictible via le choix et l'utilisation d'une distance adaptée, par exemple). Mais, dans un cadre de l'utilisation de flux de données, nous devrons en plus nous intéresser à la mise à jour incrémentale de ces classes. Des approches ont été proposées dans le cadre de mise à jour de séquences d'achat contenues dans des batches (voir Marascu et Masseglia, 2007). Les auteurs concluent, d'ailleurs, que les approches incrémentales peuvent ne pas apporter de résultats plus pertinents que dans le cadre non-incrémentales pour un temps de traitement plus conséquent. Dans notre cas, pourtant, une approche incrémentale sera incontournable au vu des objets à classer. Pour faire un parallèle avec Marascu et Masseglia (2007), nous sommes dans le cas où les séquences complètes à classer seraient présentes dans plusieurs batches (les nouveaux batches ne venant que mettre à jour les historiques des courbes panélisées).
Un autre point à aborder sera lié à la mise à jour des panels construits (soit au moment du rejet de certains clients de notre échantillon, et de l'intégration de nouveaux clients). Ce point sera plus simplement pris en compte dans le cadre d'une classification supervisée :

Quelques idées pour la classification supervisée:

Nous sommes dans le cas où nous avons comme « variable » à discriminer un ensemble de courbes, en fonction d'un certain nombre de variables décrivant le client correspondant.
Une approche possible consisterait à chercher à discriminer directement les courbes en fonction de ces autres variables. On ne s'interdit pas, à terme, d'utiliser comme variables explicatives des courbes locales de température, de nébulosité ou tout autre historique de données socio-économiques décrites sur les zones des clients échantillonnées (données issues des recensements de la population au niveau des communes, des quartiers, voire des îlots-types).
Dans un cadre « non dynamique », des traitements préalables des courbes doivent être effectués avant de construire des classifications par arbre (par exemple). Les approches développées par Ramsey et Silverman (2005) pourront être testées dans ce cas (particulièrement vu

l'hypothèse d'un travail sur des granularités à la fois variables dans le temps et entre les individus à classer que nous avons prise).
Dans le cadre « dynamique » qui nous sera offert en utilisant les flux de données, une idée serait de vérifier, à chaque arrivée de nouvelles données, l'intérêt et la stabilité de l'arbre construit ou simplement conservé à l'étape précédente. Une structure de l'arbre pourrait être conservée d'une étape à une autre, l'élagage de l'arbre pourrait être remis en question à chaque fois, ainsi que la structure de l'arbre si besoin est.
Il faudra bien entendu aborder les problèmes et contraintes exposés dans le paragraphe concernant la classification non supervisée pour justifier notre approche dans le cadre de construction de prévision, et plus particulièrement la contrainte (11).

2.3.2 Quelques idées pour les modèles de prévision et approches incrémentales:

Une fois les G_T classes construites, il nous faudra élaborer et/ou mettre à jour des modèles adaptés à chaque classe. Les trois points suivants devraient nous permettre de construire des modèles plus adaptés que ceux que nous utilisons aujourd'hui :

- Les données seront de granularités temporelles très fines (elles pourront être de granularité variable dans le temps…)
- Les données seront de nature spatio-temporelle, et devront sans doute le rester en partie à l'issue des phases de classification (de par la prise en compte de l'aspect spatial dans la classification).
- Le traitement à la volée des données des flux nous permettra d'utiliser des données récentes au niveau de chaque portefeuille étudié (ce qui est loin d'être le cas aujourd'hui).

Ainsi, le premier point nous amènera à étudier de près les travaux concernant les prévisions sur données fonctionnelles, dont les méthodes dérivées des modèles Auto-régressifs Hilbertiens (voir Bosq, 2000). D'autres travaux très récents démontrent un grand intérêt à utiliser certaines propriétés intéressantes des ondelettes pour la prévision. Sur le plan statistique, plusieurs méthodes sont disponibles : les méthodes de régression sur les coefficients d'ondelettes, les méthodes issues du spectre d'ondelettes pour généraliser les ARMA aux processus localement stationnaires et enfin les méthodes de type prévision non paramétrique mais où les similarités sont évaluées sur les coefficients d'ondelettes et non sur le signal d'origine; ainsi, des travaux très récents de Anestis Antoniadis et Theofinadis Sapatinas ont permis de développer et tester avec succès ces dernières approches, plus particulièrement sur des données de consommation électrique demi-horaire (voir Antoniadis et al., 2006). Le caractère incrémental de cette méthode (comparaison d'une fenêtre temporelle qui précède immédiatement la période de prévision – contenant entre autre, ou contenu dans les dernières données résumées issues des derniers flux transmis - avec l'ensemble des fenêtres de même largeur dans le passé pour prise en compte de leur futur immédiat dans le calcul d'une prévision par noyau) pourra être de plus, particulièrement intéressant avec notre volonté de traiter à la volée les données issues des flux. D'autres travaux sont en cours permettant de compléter ces approches en intégrant la modélisation de variables exogènes (un article est en cours de publication, sur lequel devra naturellement porter notre attention – « Functional Time Series Prediction including Exogeneous Variables »).

Il ne faudra pas mettre de côté le caractère spatial de nos données. En effet, l'influence de la température, entre autres, devrait grandement être mise en évidence et sans doute être prise en compte par des approches d'analyse et de modélisation sur données spatiales. Ainsi, la propagation de phénomènes météorologiques devrait être mise en évidence dans ce cadre.

Si nous nous orientons vers un traitement d'agrégation / désagrégation de courbes en utilisant un panel géré et administré dans un environnement de flux de données, les G_T courbes construites (au niveau des G_T classes élaborées et entretenues spécifiquement) seront entachées d'une erreur d'estimation et d'échantillonnage, quelles que soient les stratégies d'échantillonnage ou de redressement utilisées. Aussi, les données à modéliser ne seront plus des valeurs réelles indicées sur le temps, mais une suite de distributions normales, dont les variances des distributions varieront dans le temps. Dans quelle mesure devrons nous utiliser la connaissance de ces variances dans nos modélisations, à la fois dans la phase d'apprentissage de nos modèles, mais aussi dans la restitution de ces variances pour la construction d'intervalles de confiance de nos prévisions? Actuellement, l'analyse et la modélisation des données symboliques, proposées et développées entre autre dans les années 1990 et 2000 par Edwin Diday (voir Billard et Diday, 2006) paraît être une piste intéressante à aborder pour modéliser ces données un peu particulières, comparables à des objets symboliques. Des premiers travaux ont été développés dans un cadre de modélisation de séries temporelles ; ainsi des travaux récents ont été présentés par Carlos Maté Jiménez, de l'université Comillas de Madrid, au 26ème International Symposium of Forecasting concernant des modélisations par lissage exponentiel sur des séries aux valeurs de type histogramme (voir Maté, 2006). Des travaux sont en cours afin de construire des modélisations auto-régressives sur de telles données.

2.3.3 Et les modèles de prévision dans un environnement de flux de données ?

Il est bien entendu qu'un investissement autour de modèles proposés spécifiquement dans une environnement de flux de données sera mis en place. Quelques lectures sont projetées. Actuellement, 4 approches retiennent particulièrement notre attention. En premier lieu, les travaux de NN Vijayakumar, B Plale, R Ramachandran et X li (voir Vijayakamur et al., 2006) concernant des travaux de prévisions météorologique mésoéchelles (Phénomènes météorologiques intéressant une zone dont l'étendue horizontale est de l'ordre d'une centaine de kilomètres) en utilisant les flux de données avec des filtres dynamiques afin de déterminer et d'analyser certains phénomènes déclencheurs.

D'autres travaux sur des problématiques d'agrégations de données temporelles et spatio-temporelles pourront retenir notre attention (voir Zhang et al., 2003).

De même, une méthodologie, nommé AWSOM (Adaptive, Hands-off Stream-Mining) permettant, à priori, de détecter automatiquement des tendances, des phénomènes saisonniers et autres phénomènes temporels pertinents dans un cadre de flux de données en utilisant, entre autres, des décompositions en ondelettes, a fait l'objet d'articles (voir Papadimitrou et al., 2003 ainsi que Papadimitriou et al., 2004). Ils feront l'objet de lectures intéressées.

D'autres approches, concernant plutôt le cadre général du Stream-Mining seront étudiés (comme celles développées par le department of Computer Science de l'université de Stanford dans Babcock et al., 2002). Les approches concernant les régressions multiples et varia-

bles latentes abordées par analyses incrémentales des flux seront naturellement étudiées (voir Teng, 2003).

3 Conclusion

Dans cet article, nous avons mis en évidence l'intérêt d'utiliser les données issues des futurs 32 millions de compteurs communicants afin de construire des prévisions Court-termes et Moyens-termes de la consommation électrique de EDF (global, ou par portefeuilles) dans un environnement de flux de données.

L'investissement sur la recherche ou l'élaboration de techniques utilisant cet énorme volume de données « à la volée » a été évoqué. Des premières idées, provenant naturellement plus d'une logique de traitements et de modélisations « classiques » de données et de séries temporelles ont été proposées ; celles-ci sont basées sur des approches d'agrégation / désagrégation de courbes et sur l'utilisation de modélisation sur données hilbertiennes ou fonctionnelles, mais aussi sur des adaptations de techniques usuelles des sondages pour maîtriser les phases d'échantillonnage et de redressement nécessaires.

Certaines approches développées dans un cadre plus spécifique de Data-Stream et de Stream-Mining doivent faire l'objet de lectures et d'études à venir. Aussi, les liens entre les différentes approches évoquées dans ce papier devront être construits afin de les fusionner en vue de l'élaboration de méthodes et de modèles adaptés. EDF a environ 6 ans devant elle avant l'utilisation effective des données récupérées de ces compteurs. A noter qu'un laboratoire commun entre EDF et l'Ecole Nationale Supérieure des Télécommunications est mis en place en ce début 2007 afin de traiter, entre autres, de cette problématique.

Références

Antoniadis A., Paparoditis E. et Sapatinas T. (2006). *A functional wavelet-kernel approach for time series prediction.* Journal of Royal Statistical Society 68 Part 5 pages 837-857

Babcock B., Babu S., Datar M., Motwani R. et Widom J. (2002). *Models and issues in data stream systems.* ACM Symposium on Principles of Database Systems (PODS) 2002

Billard L. et Diday E. (2006). *Symbolic Data Analysis.* Conceptual Statistics and Data mining: Wiley

Bosq D. (2000). *Linear processes in function spaces. Theory and Applications.* Springer Verlag : Lectures Notes in Statistics n° 149.

Chiki R. (2007). *Répartition optimisée des pas d'échantillonnage : Application aux Courbes de charge de consommations électriques*. ECG 2007

Dessertaine A. (2006). *Sondages et séries temporelles : une application pour la prévision de la consommation électrique.* actes des journées Françaises de Statistique 2006

Dessertaine A. (2007 - à paraître). *Sampling and Data-Stream: Some ideas to built balanced sampling using auxiliary.* ISI 2007 (IPM 56 "New methods of sampling")

Deville J.C. et Särndal C.E. (1992). *Calibration estimators in survey sampling.* Journal of the American Statistical Association; 87 : 376-382

Lundström S. et Särndal C-E. et (2005). *Estimation in Surveys with Nonresponse*. Wiley

Marascu A. et Masseglia F. (2007). *Limites d'une approche incrémentale pour la segmentation de séquences dans les flux.* ECG 2007

Maté C., Arroyo J., Muñoz A.et Sarabia A. (2006). *Smoothing methods for histogram-valued time series.* 26th International Symposium on Forecasting. Santander (España). 11-14 Juin 2006

Misiti M., Misiti Y., Oppenheim G. et Poggi J.M. (1998). *Méthodes d'ondelettes en statistique : introduction et exemples.* Journal de la SFDS n° 139

Papadimitriou S., Brockwell A. et Faloutsos C. (2003). *Adaptive, Hands-Off Stream Mining*. International Conference on Very Large Data Bases (VLDB 2003)

Papadimitriou S., Brockwell A. et Faloutsos C. (2004). *Adaptive, Unsupervised Stream Mining* . VLDB Journal 2004

Papadimitriou S., Brockwell A. et Faloutsos C. (2005). *Streaming Pattern Discovery in Multiple Time-Series*. International Conference on Very Large Data Bases (VLDB 2005)

Ramsay J.O. et Silverman B.W. (2005). *Functional Data Analysis*. Springer-Verlag

Ramsay J.O. et Silverman B.W. (2002). *Applied Functional Data Analysis*. Springer-Verlag

Rao J.N.K. (2003). *Small area estimation*. Wiley

Teng W.G., Chen M.S. et Yu P.S.(2003). *A Regression-Based Temporal Pattern Mining Scheme for Data Streams*. International Conference on Very Large Data Bases (VLDB 2003)

Vijayakamur N.N., Plale B., Ramachandran R. et Li X. (2006). *Dynamic Filtering and Mining Triggers in Mesoscale Meteorology Forecasting.* IGARSS, 2006.

Zhang D., Gunopulos D., Tsotras V.J. et Seeger B. (2003). *Temporal and Spatio-Temporal Aggregations over Data Streams using Multiple Time Granularities.* Journal of Information Systems, vol. 28, no. 1-2, pages 61-84