Extraction de connaissances : réunir volumes de données et motifs significatifs

Florent Masseglia

UNIVERSITÉ DE NICE-SOPHIA ANTIPOLIS

ÉCOLE DOCTORALE STIC
SCIENCES ET TECHNOLOGIE DE L'INFORMATION ET DE LA COMMUNICATION

Mémoire présenté pour obtenir le diplôme d'
habilitation à diriger des recherches

par

Florent Masseglia
INRIA Sophia Antipolis – Méditerranée

Extraction de connaissances : réunir volumes de données et motifs significatifs.

Soutenue le 27 novembre 2009 devant le jury composé de :

Francisco Carvalho	Universidade Federal de Pernambuco	Examinateur
João Gama	University of Porto	Rapporteur
Dominique Laurent	Université de Cergy Pontoise	Rapporteur
Yves Lechevallier	Inria Rocquencourt	Examinateur
Pascal Poncelet	Université de Montpellier 2	Président
Osmar Zaiane	University of Alberta	Rapporteur

« Vite et bien, ça ne va pas ensemble ! »

Henri Masseglia (mon grand-père),
Pêcheur à l'Estaque.

Résumé

L'analyse et la fouille des données d'usages sont indissociables de la notion d'évolution dynamique. Considérons le cas des sites Web, par exemple. Le dynamisme des usages sera lié au dynamisme des pages qui les concernent. Si une page est créée, et qu'elle présente de l'intérêt pour les utilisateurs, alors elle sera consultée. Si la page n'est plus d'actualité, alors les consultations vont baisser ou disparaitre. C'est le cas, par exemple, des pages Web de conférences scientifiques qui voient des pics successifs de consultation lorsque les appels à communications sont diffusés, puis le jour de la date limite d'envoi des résumés, puis le jour de la date limite d'envoi des articles.

Dans ce mémoire d'habilitation à diriger des recherches, je propose une synthèse des travaux que j'ai dirigés ou co-dirigés, en me basant sur des extraits de publications issues de ces travaux. La première contribution concerne les difficultés d'un processus de fouille de données basé sur le support minimum[1]. Ces difficultés viennent en particulier des supports très bas, à partir desquels des connaissances utiles commencent à apparaître. Ensuite, je proposerai trois déclinaisons de cette notion d'évolution dans l'analyse des usages : l'évolution en tant que connaissance (*i.e.* des motifs qui expriment l'évolution) ; l'évolution des données (en particulier dans le traitement des flux de données) ; et l'évolution des comportements malicieux et des techniques de défense.

Les difficultés liés à un support minimum très faible : le chapitre 2 expose les motivations pour une extraction de motifs dans les usages avec des supports de plus en plus faibles. En effet, lors de nos expérimentations sur diverses sources de données d'usages, nous avons constaté que le support minimum d'extraction devait être de plus en plus bas avant de trouver des comportements pertinents. Malheureusement, cette faiblesse du support est un obstable à l'extraction en raison des temps de calculs qui en découlent. Nous avons alors proposé plusieurs solutions : l'une travaillant en aval du

[1]Le support minimum représente le pourcentage d'enregistrements dans lesquels un comportement doit se retrouver pour que celui-ci soit considéré comme fréquent.

problème en proposant d'extraire les comportements grâce à une méthode divisive, l'autre travaillant en amont en proposant de grouper les pages afin de faire artificiellement monter le support.

L'évolution en tant que connaissance : le chapitre 3 propose d'extraire des comportements qui intègrent la notion d'évolution dans leur description. Pour cela, nous analysons les données sans restriction sur la période d'analyse. Par exemple, sur une année d'exercice, un commerce pourrait concentrer son analyse sur la période des achats de Noël car c'est une saison d'activité connue. Notre but est de s'affranchir de tels *a-priori* (*i.e.* les intervalles de temps connus sur lesquels on concentre les efforts en extraction de connaissances). Nous proposons de découvrir toutes ces périodes d'activités, sans connaissances préalables sur celles-ci. Ainsi, nous pouvons extraire des comportements significatifs, associés aux intervalles de temps sur lesquels ils le sont.

L'évolution des données et les flux : le chapitre 4 présente nos travaux sur le thème de l'analyse des flux de données d'usages. Dans le contexte des flux, les données sont produites à des vitesses et dans des quantités telles que l'approximation est devenue incontournable. D'une part, les motifs extraits ne peuvent plus être exhaustifs mais, d'autre part, il faut également gérer l'historique des connaissances en raison de leur forte évolution. Dans les deux cas (*i.e.* extraction et gestion de l'historique) l'approximation est intégrée dans nos travaux et les algorithmes proposés présentent des temps d'exécution suffisants pour envisager leur utilisation dans le monde réel.

L'évolution des attaques et la sécurité : la combinaison de ces travaux sur les données évolutives, sur les flux de données et sur le faible support trouve sa place dans le domaine de la sécurité et de la détection d'intrusion. Dans le chapitre 5 nous abordons la détection d'intrusion avec une hypothèse innovante : les comportements atypiques et communs à plusieurs sites différents sont le plus souvent malicieux. Notre objectif étant de proposer une détection d'intrusion sans connaissances *a-priori* sur ces intrusions tout en minimisant le nombre de fausses alarmes. Pour réaliser ces travaux et confirmer cette hypothèse il était nécessaire de proposer une méthode d'extraction des comportements atypiques dans les flux de données et sans paramètres. Ces comportements sont alors extraits sur plusieurs sites différents et comparés entre eux.

Les travaux présentés dans ce mémoire ont été réalisés à l'Inria Sophia

Antipolis, dans l'équipe-projet AxIS[2], entre septembre 2002 et septembre 2009.

Enfin, je présenterai les conclusions et perspectives de ces travaux dans le chapitre 6, ainsi que mon CV et la liste de mes publications.

[2]L'équipe-projet AxIS est dirigée par Brigitte Trousse.

Abstract

Mining and analyzing usage data are strongly correlated to evolution. Let us consider the case of Web sites. The dynamism of usages will depend on the dynamism of related pages. If a page is created, and it is interesting for users, then it will be requested. If that page is no more appealing, the corresponding usages will drop or disappear. For instance, pages about a scientific conference will be highly requested when the call for papers is sent or when the deadlines (abstract and full papers) are reached.

In this document, I propose a synthetic presentation of research work I advised or co-advised, based on related publications. More precisely, I propose three versions of the impact of evolution in usage mining.

Chapter 2 presents the motivations of a pattern extraction with very low supports. Actually, during our experiments on various sources of usage data, we could see that the minimum support[3] had to be very low before interesting patterns are discovered. Unfortunately, a weak support means long response time. We thus proposed two solutions : the first one recursively divides the data before performing pattern extraction on its subsets and the second one proposes a generalisation of Web pages in order to simulate highest supports.

Chapter 3 proposes to extract patterns associated to periods. Therefore, evolution will be included in the extracted knowledge. To that end, we propose to analyse data without any limitation on the duration. A shop, for instance, would like to know the customers behaviours for Christmas (because this period is well known). Our goal is to discover any similar period, regardless to *a-priori* knowledge. Hence, we are able to extract significant behaviours, associated to their periods of frequency. This work on "evolution as a knowledge" showed that evolution in usage analysis is important.

[3]Minimum support is the fraction of users who should contain a pattern for it to be considered frequent.

In chapter 4 this evolution is the widest possible today, since we tackle data streams. In data streams, data arrive in a continuous stream at high speed and in large volumes. In such a context, approximation has been recognized as a key in order to extract knowledge and manage history. Approximation, in our work, has been used for pattern extraction and managing the history of these patterns.

Chapter 5 proposes a third version of evolution : that of malicious behaviours. In this work, our assumption is that malicious behaviours are outliers and will occur on more than one system. Our goal is to propose an intrusion detection based on clustering and to lower the rate of false alarms.

The research works presented in this document have been done at Inria Sophia Antipolis in the AxIS team, between september 2002 and september 2009.

Avant-propos

La définition de l'extraction de connaissance évoque l'extraction de schémas « nouveaux, non triviaux et potentiellement utiles ». L'aspect nouveau des schémas à extraire me paraît capital pour la fouille de données. J'ai voulu garder à l'esprit que les *a-priori* sur les données et sur les connaissances ne sont pas toujours productifs. Dans la plupart des cas, l'expert connait ses données et sait orienter un processus d'analyse ou d'extraction de connaissances pour en tirer les schémas les plus utiles. Mais le but de la fouille de données, depuis ses origines, est également de trouver les schémas auxquels on ne s'attendait pas forcément.

Prenons le cas des attaques sur un site Web. Il s'agit d'un domaine ou les techniques de défense et les techniques d'attaque évoluent en permanence dans un jeu du chat et de la souris. Comment l'expert peut-il prétendre savoir tout ce qu'il faut chercher pour déjouer les attaques ? D'un côté, l'expert sera indispensable pour orienter une extraction ou pour valider/invalider une hypothèse (un schéma extrait qui paraît suspect). D'un autre côté, les attaques sont par nature conçues pour être inattendues. Si il suffisait de connaître le domaine et savoir « quoi chercher » alors le problème de la détection d'intrusion serait plus facile à traiter. Malheureusement, dans la réalité, c'est souvent suite à la découverte d'une nouvelle attaque que les systèmes de protection sont mis à jour. Cette nouvelle attaque étant elle-même motivée par la découverte d'une nouvelle faille. Ainsi, une technique d'extraction cherchant des schémas en minimisant les contraintes (« quoi chercher ») a plus de chances de trouver les comportements correspondants aux attaques.

Au cours des travaux présentés dans ce mémoire, j'ai voulu augmenter l'expressivité des connaissances extraites sans y ajouter de contraintes. La motivation des approches proposées étant de laisser les données s'exprimer.

Prenons, cette fois, l'exemple d'une analyse d'un site de commerce en ligne. En guidant l'extraction avec des contraintes, on pourrait être tentés d'extraire les comportements sur la périodes des achats de Noël ou bien la période de la rentrée scolaire. Mais il existe peut-être d'autres périodes, sur lesquelles des comportements (probablement moins visibles) sont significatifs.

De manière générale, les travaux présentés dans ce mémoire veulent 1) être utiles dans une analyse du système concerné et 2) augmenter l'expressivité des connaissances extraites. L'introduction présentée dans le chapitre 1 résume les motivations de ces travaux. Ensuite, pour chaque contribution, je propose :

- La motivation de ces travaux et le contexte dans lequel ils se situent.
- Un état de l'art.
- Mes activités liés aux travaux présentés.
- Des exemples concrets de comportements découverts afin de montrer l'intérêt de ces travaux dans une analyse des usages.
- La description de la contribution.
- Une analyse d'un point de vue formel et expérimental.
- Une discussion sur les thèmes abordés.

Table des matières

Chapitre 1

Introduction

La fouille de données pour l'analyse des usages consiste à extraire des motifs ou des connaissances permettant de caractériser et de comprendre les comportements des utilisateurs d'un système ou d'une organisation. Ces comportements peuvent se trouver sous la forme d'usages, par exemple, d'un site Web, d'un système d'information médical, de distributeurs automatiques de billets, d'une bibliothèque municipale ou encore des véhicules d'un parc de location. Les connaissances extraites s'expriment en tant que **motifs d'usage** et peuvent prendre différentes formes. Parmi ces déclinaisons, citons *les objets souvent corrélés* dans les usages (par exemple deux ou trois pages d'un site Web qui sont souvent consultées ensemble par les utilisateur, ou bien des livres souvent empruntés ensemble par les abonnés d'une bibliothèque). Citons également *les classes d'usages* qui consistent à grouper des utilisateurs parce qu'ils ont des comportements similaires ou encore *les tendances* qui construisent des modèles prévisionnels permettant d'anticiper l'évolution de ces usages. Les motifs ainsi découverts sont alors exploités pour améliorer le système ou l'organisation concernés :

- le site Web peut rendre la consultation de ses pages plus intuitive ;
- la banque peut surveiller les retraits dans ses distributeurs pour éviter les fraudes ;
- la bibliothèque peut réorganiser ses rayons en fonction des consultations les plus corrélées ;
- etc.

Mes travaux à l'Inria, dans l'équipe-projet AxIS, se sont articulés autour de la fouille de données pour l'analyse des usages et en particulier ceux des sites Web. Je me suis particulièrement concentré sur la forme séquentielle de ces motifs d'usage (*i.e.* des enchaînements fréquents dans les comporte-

ments). Les motifs abordés dans ce mémoire présentent donc généralement :

- Un support, qui représente le pourcentage d'enregistrements contenant le motif.
- Des ressources (*e.g.* des pages Web) ou items/itemsets qui composent le motif.
- Une relation d'ordre temporel entre les ressources.

On parle alors de motifs séquentiels. En fixant un support minimum, l'extraction de motifs séquentiels fréquents construit l'ensemble des motifs dont le support est supérieur à ce support minimum. Considérons le site Web du centre de recherche Inria Sophia-Antipolis. Voici un exemple de motif séquentiel fréquent sur ce site :

Motif 1 *« 5% des utilisateurs*[1] *ont consulté la page d'accueil, puis la page sur les opportunités de travail ».*

Pour extraire un tel motif, il faut analyser l'ensemble des données d'usage stockées par le site. Ces données sont stockées dans un fichier log d'accès Web, qui contient les traces d'usages avec, pour chaque requête sur le site :

- L'adresse IP de la machine qui est à l'origine de la requête.
- La date de la requête (avec une précision d'une seconde).
- La ressource demandée.
- La page précédente consultée par l'utilisateur.

À partir des fichiers log on peut reconstituer des navigations (*i.e.* l'enchaînement des requêtes d'un utilisateur dans le temps). Ensuite, les comportements (exprimés par des motifs séquentiels) qui sont observés dans un nombre suffisant de navigations dans le log, sont considérés comme fréquents. Ma première observation dans l'équipe-projet AxIS, et concernant l'analyse des usages du Web, est que les motifs « fréquents » n'existent pas. On peut trouver des motifs dont la fréquence est supérieure au support minimum donné par un utilisateur, mais ce support doit être très faible avant que l'on puisse trouver des motifs exploitables. Ainsi, peut-on vraiment dire qu'un comportement commun à $0,2\%$ des utilisateurs est un comportement fréquent ? De plus, nous verrons plus loin qu'avec des supports aussi faibles les motifs sont très difficiles à extraire.

Si, en revanche, on se contente de supports élevés et de motifs semblables au motif 1, alors on obtient peu d'information sur les usages de ce site. En effet, ce type de motif est peu exploitable car il s'agit d'un motif véhiculant une information connue ou évidente (pour les concepteurs du site et pour

[1] Dans la suite de ce document, nous verrons qu'un comportement partagé par plus de 5% des utilisateurs d'un site Web est plus que fréquent.

l'analyse des usages) ne permettant pas de répondre aux besoins du site, comme par exemple rendre plus intuitive la consultation de ses pages. C'est un cas analogue à l'exemple ironique d'un processus de fouille de données appliqué sur les dossiers des patients d'une clinique permettant de découvrir que « 100% des patients enceintes sont des femmes[2] ».

Les usages les plus « attendus » sont donc les plus faciles à extraire (puisqu'ils ont un support élevé et que la complexité de leur extraction grandit quand leur support faiblit). D'une certaine manière, l'intérêt véhiculé par un schéma (*i.e.* un motif fréquent, une segmentation en plusieurs classes, une courbe de tendance, etc.) sera proportionnel à la difficulté de sa découverte. Cette difficulté a motivé mes premiers travaux dans cette équipe. En effet, un support minimum très faible entraine de nombreux calculs, des temps de réponse très longs et des résultats pléthoriques.

Cette introduction présente de manière synthétique chacune des contributions que j'ai choisi de détailler ensuite dans ce mémoire. J'appuie la description de ces travaux sur des motifs illustratifs qui sont à la fois réalistes et difficiles à extraire. Ces motifs sont parfois inspirés ou alors directement extraits des expérimentations que nous avons réalisées sur le site Web du centre de recherche Inria de Sophia-Antipolis. Les contributions présentées dans cette introduction concernent :

- l'extraction de motifs avec un support minimum très faible (section 1.1) ;
- l'extraction de motifs exprimant l'évolution (section 1.2) ;
- l'extraction de connaissances dans les flux, qui sont des sources de données fortement évolutives (section 1.3) ;
- et l'extraction de comportements malicieux en tenant compte de leur évolution (section 1.4).

[2] Quoique... si le résultat est de 99,9% alors il faut absolument trouver le patient correspondant aux 0,1% restants.

1.1 Une solution aux motifs décevants : le support très faible

Motif 2 *Parmi les utilisateurs qui ont consulté la page Web /teams/axis/Florent.Masseglia on peut en distinguer 25 % car ils ont consulté, dans l'ordre, les pages proposition_these_2010.html, publications.html et encadrement.html.*

Si on veut étudier le comportement des utilisateurs naviguant sur ma page Web afin d'en faciliter la consultation, le motif 2 est plus exploitable que celui concernant les offres d'emploi (*i.e.* le motif 1, page 2). Cependant, il existe de nombreuses pages Web de chercheurs sur le site du centre et l'extraction de tous les motifs semblables au motif 2 impose de travailler avec un support minimum très faible. Comme nous le verrons en section 1.1.1, l'extraction de motifs avec des supports très faibles pose de nombreux problèmes.

Pourtant, c'est le cumul de motifs semblables à celui-là qui peut constituer une aide à la compréhension des usages et à l'amélioration d'un site. En effet, Les utilisateurs d'un site sont très nombreux et naviguent sur des rubriques elles-même nombreuses. Les possibilités de navigations sont autant de combinaisons offertes par les chemins qu'on peut explorer sur le site. Pour améliorer sa lisibilité ou l'efficacité de sa consultation, un site ne peut pas se contenter d'exploiter les motifs les plus fréquents (comme le motif 1). Chaque partie du site sera consultée par des utilisateurs différents, avec des buts de navigation spécifiques (comme c'est le cas du motif 2). Les utilisateurs qui consultent les pages du service de communication de l'Inria Sophia-Antipolis, par exemple, ne sont pas forcément intéressés par le sujet de thèse que je propose. Il est donc nécessaire de prendre en compte tous les utilisateurs et tous les objectifs de navigation existants dans les traces d'usage si on veut proposer une analyse utile pour l'amélioration du système.

1.1.1 Difficultés liées au support minimum faible

Malheureusement, l'extraction de ces motifs, couvrant un maximum de profils utilisateurs, s'avère compliquée. Parmi les raisons qui rendent cette extraction difficile, citons :

- La grande diversité des pages sur le site. Sur un site comme celui du siège de l'INRIA, on peut compter plus de 70000 ressources filtrées (après l'étape de sélection de données), et on en compte plus de 82000 pour le site de l'unité de Sophia Antipolis. *Ce sont autant de combinaisons à explorer avec un support faible.*

- L'existence de moteurs de recherche extérieurs qui permettent à l'utilisateur d'accéder directement à la partie du site qui le concerne. *Cela diminue le nombre d'entrées dans le log d'accès Web*[3]*, mais aussi le nombre de navigations communes à plusieurs utilisateurs (i.e. le préfixe commun aux différentes navigations).*
- La représentativité de la partie du site visitée par rapport au site dans sa globalité. Le site d'une équipe de recherche peut représenter moins de 0,5 % de la globalité du site de l'INRIA. *Ainsi, chaque partie du site génère ses propres navigations, qui deviennent très nombreuses et variées, augmentant ainsi le nombre de combinaisons à explorer.*
- La représentativité des utilisateurs de cette partie du site par rapport au nombre total d'utilisateurs sur le site global. En effet, même si le site d'un équipe de l'Inria (par exemple) représente une petite fraction du site global, il peut être consulté par une majorité d'utilisateurs. En revanche, *si le site de cette équipe n'est consulté que par une très petite partie des utilisateurs (disons 0,01%) alors les usages qui lui correspondent seront difficiles à trouver et analyser.*

1.1.2 Contributions

La section 2.4 décrit nos travaux sur de nouvelles techniques de fouille pour extraire des motifs avec un support de plus en plus faible. Avec les travaux présenté dans la section 2.5, nous avons également proposé de généraliser les pages Web (sous forme de catégories) afin de faire monter la valeur du support minimum nécessaire à l'extraction.

Voici une synthèse de ces travaux.

Diviser pour découvrir

Dans la section 2.4, nous proposons une méthode destinée à la découverte des comportements fréquents « localisés ». Cette méthode est basée sur une hybridation entre l'extraction de motifs fréquents (ayant pour critère le support minimum) et la classification de ces motifs. Nous proposerons de contourner les difficultés liées aux faibles supports grâce à une méthode récursive consistant à diviser l'espace de recherche et extraire des motifs locaux. La méthode « D&D » (Divide and Discover) proposée dans la section 2.4 repose sur le principe suivant :

[3]Le fichier log d'accès Web contient toutes les requêtes des utilisateurs sur le site. Son format sera décrit plus en détails en section 2.2

1. Les motifs les plus fréquents sont découverts en spécifiant un support « raisonnable » (*i.e.* le support le plus bas possible, sans faire échouer l'extraction de motifs séquentiels).

2. Les motifs fréquents obtenus sont groupés dans des clusters.

3. On crée plusieurs sous-ensembles des données analysées. Plus précisement, on crée un sous-ensemble par cluster obtenu à l'étape 2. Dans le cas des logs d'accès Web, cela signifie un sous-log par cluster.

4. Les données d'origine sont réparties dans les sous-logs en fonction des clusters obtenus à l'étape 2. Si on considère les données log d'accès Web, on veut classer chaque navigation n lue dans le log dans un (ou plusieurs) sous-log(s). Chaque motif m de l'étape 1 est comparé à n. Si m est inclus dans n alors on insère n dans le sous-log correspondant au cluster qui contient m. Comme la navigation n peut contenir plusieurs motifs, elle peut-être insérée dans plusieurs sous-logs.

5. Le processus est répété de manière récursive sur chaque sous-log.

Ce processus est inspiré des approches par niveau. Au lieu de faire grandir un motif en testant sa fréquence par des passes successives sur les données, l'idée est de faire diminuer la taille des données pour trouver des motifs de plus en plus fréquents localement sur des sous-ensembles de ces données. Considérons le log illustré par la figure 1.1. Ces données concernent six navigations (*i.e.* n_1 à n_6). Avec un support minimum de 50%, on trouve uniquement 3 items fréquents : a, g et y (en gras dans la figure 1.1). Avec le processus D&D, on va créer trois clusters pour les motifs extraits (un par item, dans ce cas simple) et un sous-log par cluster. Par exemple, dans le sous-log correspond au cluster de l'item a, on ajoute les navigations n_1, n_2 et n_3. On peut voir à la figure 1.2 les sous-logs créés pour les clusters correspondants au motif a, au motif g et au motif y. Sur chacun de ces sous-logs, on trouve alors des motifs plus longs et plus fréquents localement. Par exemple, sur le sous-log correspondant au motif a on trouve deux motifs ayant un support de $\frac{2}{3}$: « a suivi par b » et « a suivi par g ». On remarque également que ces motifs ont un support de $\frac{1}{3}$ sur le log d'origine (figure 1.1), ce qui est inférieur au support d'origine (*i.e.* 50%).

D&D permet donc de découvrir des motifs avec un support très bas, en **divisant l'espace de recherche de manière récursive** et en découvrant les motifs de manière progressive sur des sous-ensembles des données.

n_1	**a**	b	c	
n_2	**a**	b	d	**g**
n_3	**a**	**g**	h	**y**
n_4	**g**	h	m	n
n_5	w	x	**y**	
n_6	x	**y**	z	

FIG. 1.1 – Motifs fréquents avec un support de 50%

sous-log du cluster a

n_1	**a**	**b**	c	
n_2	**a**	**b**	d	**g**
n_3	**a**	**g**	h	y

sous-log du cluster g

n_2	**a**	b	d	**g**
n_3	**a**	**g**	**h**	y
n_4	**g**	**h**	m	n

sous-log du cluster y

n_3	a	g	h	**y**
n_5	w	**x**	**y**	
n_6	**x**	**y**	z	

FIG. 1.2 – Diviser pour découvrir

Usages du Web généralisés

Nous proposons en section 2.5 une seconde solution au problème du support faible. Cette solution consiste à grouper les pages par thème, sous forme de taxonomie par exemple, afin d'obtenir un comportement plus global. Sur un portail d'ordre général (*e.g.* un portail proposant des actualités, des services de communication, etc.), on pourrait découvrir les motifs suivants :

Motif 3 *« 0,5% des utilisateurs consultent la page d'accueil puis une dépêche sur les voitures électriques, puis le cours du Dow Jones puis reviennent sur la page d'accueil avant de consulter leur mail en tant qu'abonnés »*

Motif 4 *« 0,5% des utilisateurs consultent la page d'accueil puis une dépêche sur le réchauffement climatique, puis le cours du Nasdaq puis reviennent sur la page d'accueil avant de demander le service de messagerie instantanée »*

En généralisant les pages concernées, on pourrait classer la dépêche sur les voitures électriques et celle sur le réchauffement climatique comme des pages liées à *l'environnement*. De la même manière, le cours du Dow Jones et le cours du Nasdasq peuvent être considérés comme des pages sur les *marchés financiers*. Enfin, le service de mail et le service de messagerie instantanée peuvent être considérés comme des *services de communication*.

Avec une telle généralisation, on pourrait obtenir le motif suivant :

Motif 5 *« 1% des utilisateurs consultent la page d'accueil puis une page sur le réchauffement climatique, puis une page concernant les marchés financiers, puis reviennent sur la page d'accueil avant d'utiliser un service de communication offert par le portail »*

On peut constater que le support du motif 5 est le double du support des motifs 3 et 4, ce qui correspond à notre objectif (généraliser les pages pour augmenter le support minimum des motifs extraits). Cependant, le problème d'une telle généralisation vient du temps et de l'énergie nécessaires à sa mise en place et à sa maintenance. Nous proposons alors des solutions pour automatiser cette généralisation. Pour cela nous utilisons les mots clés donnés par les utilisateurs dans un moteur de recherche afin d'accéder à ces pages. Imaginons que le portail évoqué dans cette section constate que ses pages sur le cours du Dow Jones et le cours du Nasdaq sont souvent accédés à la suite d'une recherche sur Google avec les mots clés « marchés » et « financiers ». Alors ce portail pourrait envisager d'ajouter ces mots clés dans la description de ces deux pages. De manière générale, les pages partageant les mêmes mots clés peuvent être groupées dans la même catégorie. L'idée motivant cette approche est que les internautes fournissent en permanence de grandes quantités de mots clés, en les donnant aux moteurs de recherche pour accéder aux pages qu'ils souhaitent trouver sur l'ensemble des sites disponibles. N'importe quel site peut alors connaître les mots clés qui ont permis aux internautes d'arriver sur chacune des pages qu'il héberge. Ainsi, en utilisant cette information, il est possible **d'automatiser la généralisation** et d'obtenir **des motifs aux supports plus élevés**.

1.1.3 Au delà du support minimum

Il existe un cliché répandu comparant la médecine occidentale et la médecine orientale (principalement Chinoise). Ce cliché veut que la médecine occidentale soigne les symptômes (par exemple, le mal de tête) alors que la médecine orientale soigne les causes (ce qui est à l'origine du mal de tête). En quelque sorte, extraire les motifs avec un support très faible est une façon de traiter les symptômes. Il faut maintenant traiter les causes. Mes travaux sur le thème de la faiblesse du support minimum m'ont donc conduit à des réflexions sur les raisons et les conséquences de ce phénomène. En particulier, une explication vient de l'aspect évolutif des motifs à extraire. Considérons, par exemple, les motifs relatifs à des conférences organisées par des équipes locales. Les requêtes sur ces pages seront logiquement plus nombreuses pendant la période d'actualité de ces pages. Les usages d'un site étant généra-

lement aussi dynamiques que les mises à jour de ce site, de nouvelles pages entraineront de nouveaux usages et l'obsolescence de ces pages entrainera la disparition des usages correspondants. Un mois avant une conférence, on peut s'attendre à de nombreuses requêtes sur les pages qui lui correspondent sur le site. Un an après cette conférence, on peut s'attendre à ce que les requêtes sur ces pages soient peu (ou pas) existantes. Il faut donc lier les motifs extraits à leur contexte. Il faut extraire des motifs en étant attentif au découpage temporel qui est fait des données. Un motif comme celui qui concerne une conférence sera fréquent sur la période d'activité de cette conférence. Si la période d'analyse est d'un mois, alors il y a des chances de trouver des motifs relatifs à cette conférence sur un des mois analysés. Mais si la période d'analyse est d'un an, alors le support de ces mêmes motifs risque d'être trop faible pour permettre leur découverte. Si la conférence a une période d'activité de 15 jours, à cheval sur deux mois consécutifs et que chaque mois est analysé alors ces motifs risquent d'être trouvés deux fois et de manière incomplète.

Prendre en compte l'évolution dans le processus de fouille de données et dans les motifs extraits est la motivation des travaux que j'ai ensuite conduits. L'évolution peut se décliner de plusieurs manières. Ainsi, au lieu d'extraire un motif fréquent sur un jeu de données, j'ai voulu proposer l'extraction de périodes, à l'intérieur du jeu de données, qui contiennent des motifs fréquents. Ce sujet de recherche est décrit dans la section suivante.

1.2 Des schémas qui expriment l'évolution

Motif 6 *Le jeudi 20 avril 2006, entre 07h00 et 17h00, 120 utilisateurs connectés sur le site Web du centre de recherche Inria de Sophia-Antipolis ont consulté les pages css/style.css et JoanMiro/joanmiro.html sur les pages de Christophe Berthelot de l'équipe-projet Omega.*

La première déclinaison que j'ai proposée pour cette notion d'évolution se situe à l'intérieur même des connaissances. Il s'agit d'extraire des motifs qui expriment l'aspect éphémère ou évolutionnaire des comportements extraits.

Nous avons réellement extrait le motif 6 à partir des fichiers d'accès log du site Web de l'Inria Sophia-Antipolis. L'objectif était d'extraire des périodes de taille maximale (sous-ensemble du log contigü dans le temps) ayant au moins un motif respectant le support minimum. La définition plus précise de ce problème se trouve en section 3.2. Ce motif est typique d'un intérêt lié à une période précise. Quand nous avons découvert cette période

et le motif qui lui est associé, son interprétation n'a pas été immédiate. Tout d'abord, Christophe Berthelot ne faisait plus partie de l'équipe Omega et ne pouvait donc pas nous donner d'informations sur ces pages. Nous avons alors constaté que ces pages sont dédiées à Joan Miró (artiste Catalan). Nous avons ensuite appris que Joan Miró est né le 20 avril 1893. De plus, à l'époque de ces expérimentations, la page de Christophe était classée cinquième dans les résultat de Google avec les mots clés "Joan Miro". Nous avons donc pu en conclure que pour la date anniversaire de Joan Miró, et uniquement sur cette journée, les internautes ont consulté en masse la page de Christophe.

1.2.1 Difficultés liées à l'extraction des motifs et de leurs périodes d'activité

Cependant, en explorant les traces d'usage sur une période d'un an, ce comportement a une fréquence de 0,02 %. Comme nous l'avons vu plus tôt dans cette introduction, extraire des motifs avec un support aussi faible est extrêmement difficile. En fait, nous avons identifié deux raisons principales, qui expliquent la difficulté de l'extraction d'un comportement correspondant au motif 6 :

1. Le **découpage des données** qui est fait jusqu'à présent. Ce découpage provient soit d'une décision arbitraire de produire un log tous les x jours (*e.g.* un log par mois), soit d'un désir de trouver des comportements particuliers (*e.g.* les comportements des internautes du 15 novembre au 23 décembre lors des achats de Noël). Pour comprendre ce problème du découpage des données, prenons l'exemple d'étudiants connectés lors d'une séance de TP. Imaginons que ces étudiants soient répartis en 2 groupes. Le groupe 1 était en TP le lundi 31 janvier. Le groupe 2 en revanche était en TP le mardi 1^{er} février. Chacun de ces deux groupes de TP contient 20 étudiants et la nature du TP leur demande de naviguer selon le schéma suivant : "consulter la page www-sop.inria.fr/cr/tp_accueil.html puis la page www-sop.inria.fr/cr/tp1_accueil.html puis la page www-sop.inria.fr/cr/tp1a.html". Selon le découpage des logs à raison d'un sous-log par mois, il n'y a eu pour le mois de janvier que 20 (au maximum) comportements qui respectent ce schéma. *Le fait de choisir un* **découpage arbitraire** *(ici, une analayse sur les données d'usage de chaque mois) augmente le risque de couper ce type de comportement et donc réduit les chances de le découvrir.* Les travaux que j'ai dirigés et qui sont décrits dans le chapitre 3 permettent de détecter qu'une (au moins) période dense existe et qu'elle prend place du 31 janvier jusqu'au 1^{er} février

(dates des TP). De plus, le comportement observé (tp_accueil.html → tp1_accueil.html → tp1a.html) a un support de 40 utilisateurs (les étudiants du TP). En évitant le découpage arbitraire d'une analyse par mois, le motif exprimant ce comportement sera plus cohérent (il apparaitra une seule fois et de manière complète, contrairement à une analyse issue d'un découpage arbitraire).

2. Le **support très faible** de ces motifs dans une analyse globale des données. En effet, dans la définition initiale des itemsets fréquents, l'extraction est effectuée sur la base de données toute entière (*i.e.* soit min_{supp}, le support minimum donné par l'utilisateur, les itemsets extraits doivent apparaître dans au moins $|D| \times min_{supp}$ enregistrements de D). Considérons l'exemple d'un site de commerce en ligne faisant une offre spéciale sur les DVD et les CD vierges, avec une durée limitée sur une soirée. Imaginons que ce site diffuse une publicité par e-mailing. Si on observe le comportement des clients qui profitent de cette offre, la fréquence du comportement correspondant augmente très certainement pendant les quelques heures qui suivent la campagne de publicité. D'un autre côté, *sur tout un mois d'activité du site, ce comportement sera certainement très minoritaire*. Ainsi, le défi consiste à trouver la fenêtre temporelle qui va optimiser le support de ce comportement. Les travaux que j'ai dirigés sur ce thème consistent à trouver B, un sous-sensemble contigü de D, tel que le support de ce comportement sur B est supérieur au support minimum et la taille de B est optimale. Cette connaissance (*i.e.* le comportement et la période sur laquelle il est fréquent) peut être très utile pour les décideurs qui veulent certainement découvrir ce type de comportements, et leurs périodes de fréquence, afin de les expliquer et les exploiter.

1.2.2 Contributions

La section 3.4 décrit nos travaux sur une heuristique de découpage des données permettant de découvrir des périodes denses (*i.e.* des périodes contenant des motifs d'usages fréquents). La section 3.5 présente un algorithme d'extraction de périodes qui sont calculées pour optimiser le support des motifs qu'elles contiennent.

Voici une synthèse de ces travaux.

Periodes et séquences

La section 3.4 décrit un travail préliminaire, permettant de valider les hypothèses exposées ci-dessus (*i.e.* il existe des motifs liés à des périodes particulières et pouvant générer des connaissances nouvelles). Il s'agit principalement de répondre au problème du découpage arbitraire des données en proposant un découpage alternatif. Dans ce cas, les périodes sont calculées en fonction des connexions sur le site. Pour cela, le log est analysé dans une première passe afin de reproduire les connexions et déconnexions d'utilisateurs. Ensuite, une periode de temps pendant laquelle il n'y a pas de changement (pas de connexion ni de déconnexion) sera considérée comme stable. Enfin, les motifs sont extraits sur les périodes ainsi obtenues. L'idée principale motivant ce découpage des données étant que les utilisateurs connectés ensemble peuvent avoir des buts de navigation similaires (consulter les pages du TP par exemple).

Périodes et itemsets

Dans ce deuxième cas, les périodes sont calculées pour répondre à une définition précise. Cette définition assure que :

1. Si une période est extraite alors les motifs qu'elle contient sont fréquents sur cette période.
2. Il n'existe aucune autre période plus grande sur laquelle les motifs extraits sont également fréquents.

Prenons l'exemple de la base de données D décrites par la figure 1.3. L'ensemble des motifs fréquents, avec un support minimum de 50% est donné dans le tableau de la figure 1.3 (*i.e.* F(D,1/2)). Si on considère les périodes que nous proposons de découvrir avec DeICo, le résultat est donné par la figure 1.4, où SI_n donne les itemsets fréquents de taille n sur les périodes qui leur correspondent. Par exemple, l'itemset $(a\ b\ c)$ est fréquent si on considère la période qui va de la date 7 à la date 8.

Après avoir défini ces périodes nous avons proposé l'algorithme DeICo pour extraire les périodes et les itemsets associés. DeICo est basé sur un principe Générer-Élaguer à la fois pour les itemsets et pour les périodes sur lesquelles il faut les extraire.

J'ai également co-dirigé des travaux sur la notion de tendances dans les schémas extraits. Dans ce cas, les schémas sont des motifs séquentiels et les connaissances extraites sont du type : « Dans 60% des crash du serveur d'emails, plus le nombre d'emails reçus est grand et plus la taille des emails

Tid	items	F(D,1/2)
1	**a b c**	
2	**a c** d	
3	**b** e f	
4	**c** j h	(a)
5	**a** i j	(b)
6	**b** k l	(c)
7	**a b c**	(a c)
8	m n o	
9	**a b c**	
10	**a b c**	

FIG. 1.3 – itemsets fréquents sur D avec un support de 50%

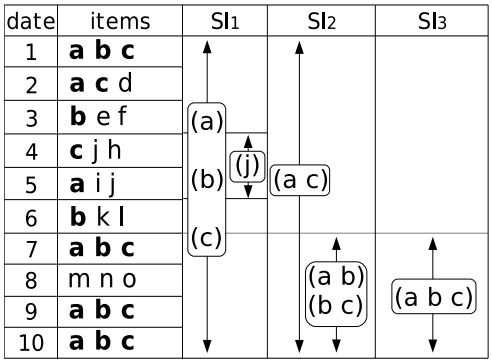

FIG. 1.4 – Itemsets compacts de D avec $\gamma = \frac{1}{2}$

est grande au temps t, plus le temps d'acheminement des messages est long. ». Je ne ferai pas une présentation exhaustive de ce travail dans ce mémoire et j'en donne les grandes lignes dans la section 3.6.1 qui propose une discussion consacrée au thème de l'évolution dans les schémas extraits.

1.2.3 Petite digression : les voitures rouges sont-elles plus chères à assurer ?

Il existe une légende urbaine disant que les voitures rouges sont plus chères à assurer[4]. L'explication de cette hausse des tarifs colorée étant que

[4]Tout au moins, je pense que c'est une légende urbaine car ma compagnie d'assurance ne m'a jamais demandé la couleur de ma voiture et je n'ai jamais rencontré quelqu'un à qui c'est arrivé (sauf si la peinture est rare et chère, comme sur les véhicules de collection,

les assureurs auraient constaté que les véhicules rouges sont *souvent* conduits par des jeunes au comportement routier sportif. Notons le terme « souvent » qui se marie particulièrement bien avec les objectifs de la fouille de données.

Imaginons alors deux cas de figure :

1. L'assureur à l'origine de cette tarification a simplement voulu valider une hypothèse. Dans son SGBD, il a isolé les voitures rouges et en a déterminé les caractéristiques (âge moyen des conducteurs, nombre d'accidents, etc.). Effrayé par les chiffres constatés, il a décidé d'augmenter les tarifs en conséquence.
2. Dans le deuxième scénario, l'assureur en question a décidé d'explorer sa base de données avec un processus d'extraction de connaissances. Comme il soupçonne un lien entre la couleur du véhicule et sa probabilité d'avoir un accident, l'assureur a ajouté la couleur dans les variables étudiées par son processus d'extraction de connaissances. Il a découvert une classe dans laquelle les accidents sont nombreux et la couleur du véhicule est rouge (et là encore, il a décidé d'augmenter les tarifs en conséquence).

Dans le premier scénario, il s'agit de valider une hypothèse. Il n'y a donc aucune extraction de connaissances dans ce cas. Dans le deuxème scénario, il s'agit d'extraire des connaissances avec un processus biaisé par un *a-priori* (*i.e.* « la couleur du véhicule a certainement une influence sur les chances d'avoir un accident »). Cependant, le but de l'extraction de connaissances est justement de **découvrir** des connaissances. À mon avis, il est dommage de limiter ce processus à des sous-ensembles des données construits sur la base d'hypothèses. Et si il existait un critère, différent de la couleur, auquel l'assureur n'a pas pensé ? Et si, en utilisant toutes les variables existantes, il trouvait une corrélation importante, montrant qu'une baisse des tarifs sur une certaine catégorie de véhicules serait judicieuse ?

De manière générale, mes travaux sont motivés par l'inhibition des contraintes ou des connaissances *a-priori* dans le processus de fouille. Je crois à la possibilité de découvrir des corrélations surprenantes auxquelles même l'expert ne s'attend pas forcément (mais qu'il peut expliquer et exploiter). Les travaux exposés dans le chapitre 3 sont basés sur cette motivation.

---

et qu'en cas d'accident l'assureur craint que la note monte très haut).

1.3 L'évolution des données et le traitement des flux

La fréquentation du site a augmenté significativement et il n'est plus possible de stocker les traces d'usage en raison de leur volume et de leur vitesse de production. Comment savoir si le motif 2 (page 4) et le motif 6 (page 9) existent toujours ? Comment gérer leur évolution dans le temps ?

Quand j'étais enfant et que je voulais exécuter très rapidement un geste alors que je n'avais pas encore appris à le maitriser, mon grand-père me disait inévitablement : « Petit... vite et bien, ça va pas ensemble ! ». Autrement dit, je pouvais choisir de continuer à cette vitesse pour un résultat moyen ou bien accepter le fait qu'un bon résultat demande plus de temps et d'attention. Les caractéristiques des flux de données imposent d'optimiser un compromis similaire. Un flux produit de grandes quantités de données de manière potentiellement infinie et à une grande vitesse. Cette expression « grande vitesse » peut évidemment paraître imprécise (ce n'est pas une définition au sens académique du terme). Généralement, on entend par « grande vitesse » le fait que les outils (machines et méthodes) actuels ne sont pas capables de traiter ces données en temps réel, à cause de cette vitesse de production.

Il s'agit donc d'une *deuxième déclinaison de la notion l'évolution* : celle des données analysées. Dans le contexte des flux, les données sont fortement évolutives car elles sont produites puis supprimées, le tout avec une grande rapidité.

Un flux ne peut pas être bloqué. Il est donc impossible d'utiliser certains traitements classiques comme les opérations de jointure des algorithmes fonctionnant par niveau. Il est généralement admis que, pour traiter un flux, il faut trouver le meilleur compromis entre le temps d'exécution du processus d'extraction et la qualité des résultats produits par ce processus. Pour optimiser ce compromis, il faut d'abord accepter un degré d'approximation dans les résultats obtenus par le processus d'extraction.

1.3.1 L'approximation dans les flux

Prenons le cas de l'échantillonnage. Cette technique permet d'extraire des connaissances sur un sous-ensemble des données tout en garantissant que ces connaissances sont fiables. Dans ce cas, « fiables » signifie que ces connais-

sances s'approchent des connaissances qui seraient obtenues en travaillant sur la totalité des données, avec une marge d'erreur maitrisée. Évidemment, la taille de l'échantillon est le principal facteur d'influence sur la marge d'erreur. L'échantillonnage est donc un des meilleurs candidats à l'extraction de connaissances dans les flux. En effet, on peut choisir la taille de l'échantillon en fonction de la vitesse d'exécution qu'impose le flux et accepter la marge d'erreur qui en résulte. Cependant, selon la nature de l'application, on peut trouver des défauts à cette approche. Par exemple, si l'objectif est de trouver des objets rares dans le flux (des anomalies) alors la marge d'erreur devient plus grande (la probabilité de traiter un événement rare par échantillonnage est plus faible que celle de traiter un événement fréquent). Dans ce cas, on voudrait privilégier une méthode qui tient compte de tous les enregistrements, mais qui utilise l'approximation dans le traitement des données (par opposition à une approximation dans la sélection des données traitées).

Prenons alors le cas de la classification non-supervisée. Il s'agit de grouper les objets par similitude. Dans ce cas, plus les comparaisons entre les objets sont nombreuses, meilleure est la qualité espérée. L'approximation peut alors être utilisée dans de nombreuses étapes du processus de classification : la distance utilisée pour comparer les objets (qui peut devenir une mesure de similitude pour accélérer les calculs), le processus de regroupement des objets (qui peut aller du hiérarchique à l'agglomératif naïf), etc. En choisissant une mesure de similitude approximative mais rapide et un processus de regroupement naïf mais qui satisfait les contraintes du flux, on peut traiter toutes les données du flux et en proposer une classification exhaustive. Cette classification sera bien sûr approximative, mais elle sera tout de même une base pour trouver les événements rares, ce qui posait problème avec l'échantillonnage (on peut, par exemple, envisager les techniques de détection d'anomalies basées sur la classification non-supervisée).

Enfin, l'approximation doit également intervenir dans l'observation à long terme du flux. Les résumés constitués sur les flux ou bien les connaissances extraites à partir de ces flux doivent être historisées. D'un autre côté, étant donné que les flux produisent des données de manière continue et potentiellement infinie, il n'est pas envisageable que ces historiques soient exhaustifs. Une compression doit être mise en place et cette compression ne peut pas être sans perte. Les premiers modèles imaginés dans ce domaine [GHP+03, TCY03, CL03] ont estimé que les événements les plus anciens doivent être oubliés en premier. En d'autres termes « plus c'est vieux, moins c'est important ».

1.3.2 Contributions

La section 4.5 présente notre technique d'extraction de motifs séquentiels dans les flux de données basée sur un découpage du flux en paquets de taille fixe et un clustering des séquences de chaque paquet pour en extraire des motifs. Dans la section 4.6 nous présentons une nouvelle stratégie de gestion de l'historique des motifs extraits.

Voici une synthèse de ces contributions.

Extraction de motifs séquentiels approximatifs dans les flux

(a)	(b)		(d)
(a)		(c)	(d)
(a :2)	(b :1	c :1)	(d :2)

FIG. 1.5 – Principe de l'alignement de deux séquences

La section 4.5 décrit la façon dont nous avons introduit l'approximation dans l'extraction de motifs séquentiels à partir des flux de données. Cette approximation est basée sur une technique d'alignement de séquences, qui permet de trouver un motif séquentiel représentatif d'un ensemble de séquences. Le principe de l'alignement est illustré par la figure 1.5. On peut voir dans cette figure deux séquences : « a suivi par b suivi par d » ainsi que « a suivi par c suivi par d ». Ces deux séquences trouvent un alignement exprimé par la dernière séquence de la figure 1.5. Il s'agit d'un motif qui représente l'ensemble des séquences alignées. Ce motif peut se lire : « a suivi par (b ou c) suivi par d ». Ce motif contient également des compteurs associés à chaque élément. Si l'objectif est d'obtenir un motif représentatif de ce qui est le plus fréquent dans les séquences alignées, on obtiendrait « a suivi par d » en utilisant la valeur la plus élevée qui soit dans la séquence alignée (*i.e.* la valeur 2).

Le principe général que nous proposons pour extraire les motifs séquentiels approximatifs dans les flux de navigations est le suivant :

1. Tout d'abord, nous proposons de découper le flux en paquets de taille fixe.
2. Pour chaque paquet, les séquences de navigation qu'il contient sont regroupées dans des classes.
3. À la fin du traitement du paquet, pour chaque classe, les séquences de navigation qu'il contient sont alignées afin d'en extraire un motif

séquentiel représentatif.

La difficulté de ce principe vient de la classification des séquences de navigation. Pour accélérer un processus de classification non supervisé, il est courant de définir un centre pour les classes. Cela permet de comparer les objets à classer avec les centres des classes existantes, plutôt qu'avec tous les autres objets. Dans le premier cas (comparaison aux centres) la complexité est de $O(n.m)$ avec n le nombre d'objets et m le nombre de classes. Dans le deuxième cas elle est de $O(n^2)$. Étant donné que, généralement, m est très inférieur à n, le premier cas doit être privilégié pour améliorer les temps de calcul. Dans notre cas, un centre tout désigné pour les classes se trouve dans l'alignement des séquences de cette classe.

Malheureusement, l'alignement ne se prête pas à une arrivée des séquences par ordre aléatoire. Ce qui le rend incompatible avec un traitement incrémental des séquences. Ainsi, on ne peut pas mettre à jour le centre des classes quand de nouvelles séquences sont ajoutées. Dans la section 4.5 nous proposons une technique pour contourner cette incompatibilité et permettre une **classification des séquences de navigations avec une approche centroïde basée sur l'alignement**. À la fin du traitement de chaque paquet, les centres de chaque classe sont considérés comme des motifs séquentiels extrait depuis le flux.

Gestion de l'historique des motifs extraits

Pour la gestion de l'historique, le modèle proposé est totalement nouveau et consiste à privilégier les événements saillants plutôt que les événements récents. Nous proposons donc d'opposer le modèle « plus c'est marquant, plus c'est intéressant » au modèle traditionnel qui veut que « plus c'est ancien, moins c'est intéressant ». Pour cela, les historiques à gérer sont mis en concurrence pour obtenir de l'espace mémoire. Les historiques les mieux représentés vont libérer de la mémoire alors que ceux qui présentent un taux d'erreur plus élevé dans leur réprésentation seront plus gourmands en mémoire. L'approche est comparée aux modèles de vieillissement (*decaying factor*) qui sont les plus répandus dans ce domaine. Ces travaux sont décrits dans la section 4.6.

1.4 La sécurité des systèmes d'information et l'évolution des attaques

Motif 7 *Il existe un groupe de 280 utilisateurs ayant consulté, pour 85% d'entre eux, les pages (dans cet ordre) :*
- *scripts/root.exe*
- *c/winnt/system32/cmd.exe*
- *..%255c../..%255c../winnt/system32/ cmd.exe*
- *..%255c../..%255c/..%c1%1c../..%c1%1c../winnt/system32/cmd.exe*
- *et winnt/system32/cmd.exe (3 fois)*

De plus, ce comportement apparait également sur le site de l'IRISA avec des caractéristiques similaires.

Quand nous avons extrait le motif 7 lors de nos expérimentations sur le site Web de l'Inria Sophia, son interprétation a demandé une « enquête ». Nous avons alors consulté les responsables du site Web afin de leur soumettre ce motif et leur demander si ils avaient une idée sur son origine. Il s'agit en fait d'une tentative d'intrusion. L'attaque consiste à exploiter le bug unicode[5] sur un serveur tourant sous Microsoft Windows. Sachant que le site Web de l'Inria Sophia n'est pas géré sous un système Windows, on peut se demander pourquoi cette tentative (vouée à l'échec) a été lancée sur notre site Web. L'explication vient du naturel « partageur » des attaquants. Souvent, quand un attaquant trouve une faille sur un système, il génère un script pour exploiter cette faille de manière automatique sur d'autre sites (il veut automatiser l'attaque). Ensuite, ce script sera utilisé sur de nombreux sites avec l'espoir que l'un deux sera assez fragile pour y céder. Puis le script est partagé avec d'autres attaquants. Eux-même vont lancer le script sur de nombreux sites et ils vont parfois le modifier pour l'améliorer. Ainsi, on retrouve des comportements très similaires sur un petit groupe (celui des attaquants qui ont testé le script sur le site), avec un support faible et sur plusieurs sites (par exemple sur les sites Web de l'Inria Sophia et de l'IRISA).

Cette troisième (et dernière) déclinaison de l'évolution concerne la lutte permanente qui oppose les attaquants et les défenseurs des systèmes. Les techniques d'attaques évoluent pour contrer les défenses actuelles et les techniques de défenses évoluent pour prendre en compte les nouvelles attaques,

[5] L'exploitation du bug unicode consiste à remplacer certains caractères par un codage différent. Par exemple le caractère espace devient '%20'. Le but est d'échaper aux détecteurs de signatures en essayant de « déguiser » le contenu de l'attaque. Ce bug est maintenant corrigé, mais les attaques qui tentent de l'exploiter sont encore nombreuses.

ou pour les anticiper. L'évolution se situe également au niveau des données. Les données d'usages étant de plus en plus produites sous forme de flux, les techniques de détection doivent s'adapter à ces caractéristiques.

1.4.1 Les systèmes de détection d'intrusion

Les systèmes de détection d'intrusions, ou IDS (Intrusion Detection Systems), existants peuvent se diviser en deux catégories : ceux basés sur les signatures et ceux basés sur les anomalies. Dans le cas des signatures, l'IDS dispose d'une base de signatures qui correspondent à des attaques. Par exemple, la requête *http ://www.site.fr/annuaire.php ?nom="../../etc/passwd"* doit être bloquée[6] car elle contient la signature *etc/passwd* qui est typique d'une tentative de récupérer le fichier des mots de passe sur le système. Dans le cas des anomalies, on peut encore diviser les IDS en deux catégories : les semi-supervisés et les non-supervisés. Un IDS semi-supervisé dispose d'une classification des comportements normaux. Si un comportement ne correspond pas à une de ces classes alors il est considéré comme déviant et doit déclencher une alarme. Proposer une liste des comportements normaux est un travail fastidieux qui demande des mises à jour constantes et la mobilisation de nombreux experts du système (qui comprennent les usages et peuvent dire lesquels sont normaux ou pas). Pour éviter cela, un IDS non-supervisé ne dispose d'aucune connaissance préalable et doit distinguer de manière autonome les comportements normaux des anormaux. Généralement, cela se fait par une étape de clustering permettant de grouper les comportements par similitude. Ensuite, les comportements éloignés des autres (les comportements isolés, n'appartenant à aucune classe ou bien les comportements groupés dans des classes de très petite taille par exemple) sont considérés comme anormaux.

Pour mieux comprendre le problème de la lutte attaquants/défenseurs ainsi que les avantages et défauts des différents types d'IDS, considérons les deux scénarios suivants.

Scénario avec un IDS basé sur les signatures :

1. L'attaquant envoie la requête :
   http ://www.site.fr/annuaire.php ?nom="../../etc/passwd"
   Le script est mal protégé et renvoie le fichier des mots de passe. L'intrusion est un succès.

[6] Il s'agit d'une tentative typique, basique et désormais innoffensive. L'objectif est de détourner un script PHP en espérant qu'il soit assez peu protégé pour aller effectivement chercher le fichier passé en argument et le renvoyer au demandeur.

2. L'intrusion est découverte et le motif *"../../etc/passwd"* est ajouté à la liste de signatures.
3. La même requête est envoyée et l'intrusion est un échec.
4. Un attaquant est assez rusé pour trouver le bug unicode. En d'autres termes, la chaîne de caractères "*../*" est interdite mais son codage unicode "*..%c0%af*" ne l'est pas (en unicode le caractère "/" s'écrit "*%c0%af*").
5. La requête suivante est envoyée :
   http ://www.site.fr/annuaire.php ?nom="..%c0%af..%c0%afetc/passwd"
   Elle ne correspond pas à une attaque connue et passe le contrôle de sécurité. Elle est transmise au décodeur unicode qui remet en place les caractères correspondants. L'attaquant récupère le fichier des mots de passe.
6. L'intrusion est découverte, le bug unicode est mis à jour et la signature est ajoutée à la base.
7. La compétition attaquants/défenseurs continue[7].

Scénario avec un IDS basé sur les anomalies :

1. L'attaquant envoie la requête :
   http ://www.site.fr/annuaire.php ?nom="../../etc/passwd"
   La requête est rare ou atypique et déclenche une alarme.
2. Un nouvel employé (Dupont) vient d'arriver et un utilisateur bien intentionné envoie la requête :
   http ://www.site.fr/annuaire.php ?nom="Dupont"
   Cette requête est atypique (Dupont vient d'arriver dans la société) et déclenche une alarme.
3. La requête suivante est envoyée (bug unicode) :
   http ://www.site.fr/annuaire.php ?nom="..%c0%af..%c0%afetc/passwd"
   La requête est rare ou atypique et déclenche une alarme.

Comme on peut le voir dans les deux scénarios précédents, chaque système a ses défauts. Les IDS à base de signatures sont incapables de défendre le système contre les nouvelles attaques (puisque leurs signatures sont encore inconnues). Ce n'est qu'après la découverte de cette faille (généralement révélée par une attaque) que les signatures sont mises à jour pour la sécuriser. Les IDS à base d'anomalies sont trop sensibles et déclenchent des fausses

[7] Aujourd'hui encore, les grands éditeurs de solution de défense proposent des mises à jours quotidiennes. Ces mises à jours sont généralement compilées trois fois toutes les 24 heures dans des agences situées sur 3 continents différents qui se passent le relais.

alarmes (« Dupont ») en trop grand nombre. Généralement, les IDS à base de signatures sont les plus répandus. La raison principale étant que la détection d'anomalie en tant qu'IDS reste confinée au monde académique à cause de ses trop nombreuses alarmes qui rendent ce principe peu exploitable dans le monde industriel.

1.4.2 Contributions

La section 5.4 présente nos travaux sur la détection d'anomalies communes à plusieurs sites. La section 5.5 présente une nouvelle technique non paramétrée de détection des anomalies dans un flux.

Voici une synthèse de ces travaux.

Détection d'Outliers Communs

La méthode DOC (Détection d'Outliers Communs), présentée en section 5.4, se base sur une analyse des usages observés sur plusieurs systèmes partenaires. Notre idée principale est que les nouveaux usages sont généralement liés au contexte du système d'information sur lequel ils apparaissent (donc ils ne sont pas sensés être partagés par plusieurs systèmes). D'un autre côté, **quand une faille est trouvée sur un système, l'attaquant voudra utiliser cette faille sur le plus grand nombre possible de systèmes**. Ainsi, une nouvelle anomalie qui serait partagée par deux (ou plus) systèmes partenaires n'est probablement pas dûe à de nouveaux usages mais plutôt à un comportement malveillant. Considérons A_x, une anomalie détectée dans les usages du site Web S_1 et correspondant à une requête PHP sur l'annuaire pour chercher un nouvel employé : Jean Dupont, qui travaille dans le bureau 204, étage 2, département R&D. La requête sera de la forme : `staff.php?FName=Jean\&LName=Dupont\&room=204\&floor=2\&Dpt=RD`. Cette nouvelle requête, dûe au recrutement récent de Jean Dupont dans ce département, ne devrait pas être considérée comme une attaque. D'un autre côté, considérons A_y, une anomalie correspondant à une véritable intrusion. A_y sera basée sur une faille du système (par exemple une vulnérabilité dans un script PHP) et pourrait, par exemple, être de la forme : `staff.php?path=../etc/passwd%00`. On peut voir dans cette requête que les paramêtres ne sont pas liés aux données accédées par ce script PHP, mais plutôt à une faille découverte dans le script *staff*. Imaginons maintenant que ce script soit produit par une société de production de logiciels et que plusieurs compagnies l'aient acheté. Alors l'exploitation de cette faille serait certainement répétée sur les sites des compagnies concernées, géné-

rant des anomalies communes ayant pour paramètre : `../etc/passwd%00`. Cette première contribution au domaine de la fouille de données pour la sécurité est une méthode prenant un seul paramètre : n, la limite haute sur le nombre d'alarmes désirées. Ensuite, sur la base de l'analyse des usages sur les différents partenaires, notre méthode détectera n anomalies communes. De telles anomalies partagées ayant de fortes chances d'être des attaques, elles permettront de déclencher des alarmes.

Détection d'Outliers par les Ondelettes

À notre connaissance, les méthodes existantes pour la détection d'outliers reposent toujours sur un paramètre qui situe le degré d'atypicité au delà duquel les enregistrements doivent être considérés comme inhabituels. Par exemple, dans [KN98], un outlier est un objet tel qu'une fraction des objets dans les données est éloignée de cet outlier d'une distance au moins égale à celle spécifiée par l'utilisateur. Dans [FZFW06], les auteurs proposent une méthode de clustering sans paramètre et la détection d'outliers est basée sur une valeur k utilisée dans la détection des top-k outliers. Nous proposons DOO (Détection d'Outliers par les Ondelettes), une méthode sans paramètre destinée à l'extraction automatique d'outliers dans les résultats d'un algorithme de clustering. La différence avec les travaux existants réside dans **une technique de séparation automatique des valeurs en deux ensembles correspondants au segments (clusters) et aux outliers (anomalies)**. Notre méthode s'adapte à tous les résultats d'un algorithme de segmentation et toutes les caractéristiques peuvent être utilisées (distances entre objets [KN98], densité [BKNS00, PKGF03] ou taille des clusters [JTS01, SZ02]). Pour cela, nous proposons de considérer la distribution des clusters en fonction de leur taille, après les avoir classés par taille croissante. Cette distribution sera alors découpée en deux sous-ensembles correspondants aux anomalies (« petits » clusters) et aux clusters normaux, grâce aux ondelettes de Haar. Nous calculons en effet la décomposition de cette distribution et nous gardons les deux coefficients les plus significatifs. Les deux plateaux obtenus permettent alors de distinguer les petits clusters sans faire intervenir de paramètre utilisateur et de façon auto-adaptative.

1.5 Situation par rapport à mes travaux de thèse

Lors de mes travaux de thèse [Mas02] je me suis concentré sur l'extraction de motifs séquentiels dans les bases de données. Ma première préoccupation étant l'efficacité des méthodes d'extraction, j'ai proposé deux algo-

rithmes destinés à extraire les motifs grâce à une nouvelle structure et un nouveau schéma algorithmique (PSP [MCP98a]). J'ai ensuite analysé l'impact du contexte dans les différents environnements de la fouille de données. Parmi ces éléments de contexte, citons l'aspect incrémental [MPT99] ou encore l'aspect distribué des calculs [MTP01].

Les travaux que j'ai menés à l'Inria m'ont conduit à travailler dans des domaines auxquels je n'avais pas encore contribué lors de ma thèse. Parmi ces domaines, je citerais principalement :

- **L'analyse des usages** et particulièrement les problèmes liés à :
  - **La faiblesse du support** permettant d'extraire des motifs utiles.
  - **L'extraction de motifs exprimant** des connaissances supplémentaires comme **les périodes** sur lesquelles ils sont valides.
- **La classification non supervisée** : pour diviser les données et extraire les motifs à faible support, pour obtenir une généralisation des pages Web, pour proposer une segmentation des usages dans les flux ou encore pour détecter les classes atypiques et en déduire des intrusions.
- **La fouille de flux de données**. Lors de ma thèse j'avais abordé l'aspect incrémental de la fouille de données, mais les flux ont des contraintes supplémentaires qui rendent les deux domaines bien distincts.
- **La détection d'intrusions et d'anomalies**.
- Et d'autres domaines non abordés dans ce mémoire tels que la classification de documents XML, l'extraction de motifs exprimant les tendances, la logique floue dans la fouille de données ou encore la confidentialité des échanges dans la fouille de données.

1.6 Organisation du mémoire

Le chapitre 2 présente nos travaux relatifs au problème du faible support des motifs à extraire. Nous y répondons avec 1) une technique d'extraction par divisions récursives et 2) une généralisation des objets à analyser.

Le chapitre 3 est une suite naturelle de ces travaux avec l'ambition de mieux comprendre les raisons de cette faiblesse du support. L'évolution sera mise en avant comme étant une des explications et sera intégrée dans les connaissances à extraire.

Le chapitre 4 prend en compte une forme d'évolution extrême avec les flux de données. Nous y verrons les difficulté liées à l'extraction dans de tels contextes mais aussi les défis posés par la gestion de l'historique des

connaissances extraites.

Dans le chapitre 5, l'évolution concerne les attaques sur les systèmes d'informations. En effet, ces attaques évoluent en permanence, obligeant les systèmes de sécurité à tenir le rythme. Afin de résister à ces évolutions nous proposons quelques solutions.

Le chapitre 6 présente les conclusions et perspectives de ces travaux. Enfin, le chapitre 7 résume l'ensemble de mes activités.

Chapitre 2

À la loupe : quand le support devient de plus en plus faible

Les travaux présentés dans ce chapitre ont pour motivation la difficulté d'extraction de motifs utiles dans les traces d'usage du Web. En effet, l'analyse des usages d'un site Web à partir d'une extraction de motifs est souvent limitée par le faible support de ces motifs. Cela est dû principalement à la grande diversité des pages et des comportements.

Dans un premier temps, notre objectif est de découvrir des comportements parfois minoritaires mais dont la cohérence les rend trop intéressants pour ne pas les considérer (comme les attaques pirates sur un site ou les personnes qui consultent la présentation d'une équipe de recherche). **En nous basant sur une classification des motifs séquentiels obtenus sur le log, nous proposons une division récursive du problème**. Nos expérimentations montrent l'obtention des motifs visés, mais aussi que leur découverte par un processus classique est impossible car elle demande de spécifier un support trop faible (jusqu'à 0,006 %). En effet l'intégralité des sessions présente une telle diversité de comportements que les plus minoritaires (sortes de « niches ») sont à la fois nombreux et particulièrement difficiles à isoler. Ces travaux sont décrits dans la section 2.4. Ils sont décrits plus en détails dans le Chapitre 3 de la thèse de Doru Tanasa [Tan05].

Dans un deuxième temps, nous proposons de regrouper la plupart des pages dans différentes catégories lors d'un pré-traitement. Travailler sur ces catégories, plutôt que sur les URLs, permet de faire émerger certains comportements de manière "générique". Nous présentons une méthodologie ori-

ginale d'analyse des usages du Web à partir d'une généralisation des URLs. L'originalité de notre approche consiste à utiliser les usages comme source d'information pour généraliser les pages. En particulier, **nous considérons les mots clés donnés par les utilisateurs aux moteurs de recherche avant d'arriver sur les pages.** Quand un mot clé est souvent utilisé pour trouver une page, alors on peut envisager de l'utiliser pour caractériser cette page. Quand plusieurs pages peuvent être caractérisées par le même mot clé alors on peut envisager de les généraliser grâce à ce mot clé. Nous présentons ensuite une méthode et des expérimentations relatives à une généralisation des URLs basée sur des informations relatives aux accès aux pages : celle-ci permet de mettre en avant les changements de support des motifs extraits selon qu'ils sont obtenus avec ou sans généralisation. Ces travaux sont décrits dans la section 2.5.

2.1 Activités

2.1.1 Encadrement

- **Doctorant** : Lors de ces travaux, j'ai participé à l'encadrement de Doru Tanasa (directrice de thèse : Brigitte Trousse). Les travaux correspondants sont décrits dans la section 2.4. Doru Tanasa a soutenu sa thèse le 3 juin 2005. Titre du mémoire de thèse : « Web Usage Mining : Contributions to Intersites Logs Preprocessing and Sequential Pattern Extraction with Low Support ». Ma participation à l'encadrement de la thèse de Doru Tanasa (15%) portait sur le thème des méthodes hybrides d'extraction de motifs séquentiels pour le faible support.
- **Master** : j'ai co-encadré le travail de Master Recherche 2ème année de Sofiane Sellah en 2005 (co-encadrante : Brigitte Trousse). Les travaux correspondants sont utilisés dans un travail collectif décrit en section 2.5. Le stage de Sofiane Sellah portait sur la généralisation des pages Web pour l'extraction de motifs séquentiels d'usages. Titre du mémoire de stage de master : « WUM : Extraction de motifs séquentiels selon plusieurs points de vue » (juillet 2005). Taux de participation à l'encadrement de Sofiane Sellah : 50%.

2.1.2 Publications

Les travaux décrits dans ce chapitre ont donné lieu à des publications dans un ouvrage international, dans une conférence internationale et dans

des conférences nationales :

- Doru Tanasa, Florent Masseglia and Brigitte Trousse. « Mining Generalized Web Data for Discovering Usage Patterns ». In Encyclopedia of Data Warehousing and Mining - 2nd Edition. Information Science Publishing. ISBN : 978-1-60566-010-3. August 2008.
- F. Masseglia, D. Tanasa, and B. Trousse. « Web Usage Mining : Sequential Pattern Extraction with a Very Low Support ». In Advanced Web Technologies and Applications : 6th Asia-Pacific Web Conference , APWeb 2004, Hangzhou, China, April 14-17, 2004. Proceedings, volume 3007 of LNCS, pages 513-522, 2004. Springer-Verlag.
- Doru Tanasa, Florent Masseglia, Brigitte Trousse. « GWUM : une généralisation des pages Web guidée par les usages », INFORSID Informatique des organisations et systèmes d'information et de décision, Hammamet, Tunisie, June 2006.
- D. Tanasa, B. Trousse et Florent Masseglia « Classer pour Découvrir : une nouvelle méthode d'analyse du comportement de tous les utilisateurs d'un site Web ». In Extraction et Gestion des Connaissances EGC'2004, pp 549-560, January 2004.
- F. Masseglia and D. Tanasa, and B. Trousse « Diviser pour Découvrir : une Méthode d'Analyse du Comportement de Tous les Utilisateurs d'un Site » In Actes des 19ièmes Journées Bases de Données Avancées (BDA'03), Lyon, France, Octobre 2003.

2.2 Définitions

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 [AIS93] et [AS95]. Dans [AIS93], le problème de la recherche de règles d'association dans de grandes bases de données est défini de la manière suivante.

Définition 1 Soit $I = \{i_1, i_2, ..., i_m\}$, un ensemble de m achats (*items*). Soit $D = \{t_1, t_2, ...t_n\}$, un ensemble de n transactions ; chacune possède un unique identificateur appelé TID et porte sur un ensemble d'items (*itemset*) I. I est appelé un *k-itemset* où k représente le nombre d'éléments de I. Une transaction $t \in D$ contient un itemset I si et seulement si $I \subseteq t$. Le *support* d'un itemset I est le pourcentage de transaction dans D contenant

I : $supp(I) = \|\{t \in D \mid I \subseteq t\}\|/\|\{t \in D\}\|$. Une règle d'association est une implication conditionnelle entre les itemsets, $I_1 \Rightarrow I_2$ où les itemsets I_1, $I_2 \subset I$ et $I_1 \cap I_2 = \emptyset$. La *confiance* d'une règle d'association $r : I_1 \Rightarrow I_2$ est la probabilité conditionnelle qu'une transaction contienne I_2 étant donné qu'elle contient I_1. Le support d'une règle d'association est défini par $supp(r) = supp(I_1 \cup I_2)$. Etant donné deux paramètres spécifiés par l'utilisateur, *minsupp* et *minconfiance*, le problème de la recherche de règles d'association dans une base de données D consiste à rechercher l'ensemble des itemsets fréquents dans D, *i.e.* tous les itemsets dont le support est supérieur ou égal à *minsupp*. Puis, à partir de cet ensemble, générer toutes les règles d'association dont la confiance est supérieure à *minconfiance*.

Pour étendre la problématique précédente à la prise en compte du temps des transactions, les mêmes auteurs ont proposé dans [AS95] la notion de séquence définie de la manière suivante.

Définition 2 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 $(i_1 i_2 \ldots i_k)$ 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 $< s_1 s_2 \ldots s_n >$ 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, $T_2, \ldots,$ 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 $< itemset(T_1)\ itemset(T_2) \ldots\ itemset(T_n) >$.

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 3 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*.

Adapter la problématique des motifs séquentiels à l'analyse des usages du Web

Ce paragraphe propose de reprendre les concepts essentiels d'un processus de Web Usage Mining, afin de présenter de façon synthétique les procédés mis en œuvre lors de l'analyse du comportement des utilisateurs d'un site web. Les principes généraux sont similaires à ceux du processus d'extraction de connaissances exposés dans [FPSSU96]. La démarche se décompose en trois phases principales. Tout d'abord, à partir d'un fichier de données brutes, un prétraitement est nécessaire pour éliminer les informations inutiles. Dans la deuxième phase, à partir des données transformées, des algorithmes de data mining sont utilisés pour extraire les itemsets ou les séquences fréquents. Enfin, l'exploitation par l'utilisateur des résultats obtenus est facilitée par un outil de requête et de visualisation. Les données brutes sont collectées dans des fichiers access log des serveurs web. Une entrée dans le fichier access log est automatiquement ajoutée chaque fois qu'une requête pour une ressource atteint le serveur web (*demon http*). Les fichiers access log peuvent varier selon les systèmes qui hébergent le serveur, mais présentent tous en commun trois champs : l'adresse du demandeur, l'URL demandée et la date à laquelle cette demande a eu lieu. Parmi ces différents types de fichiers, nous avons retenu le format CLF spécifié par le CERN et la NCSA [Con98] pour les logs HTTP, une entrée contient des enregistrements formés de 7 champs séparés par des espaces :

```
host user authuser [date:time] "request" status bytes
```

La figure 2.1 illustre un extrait de fichier access log du serveur web de l'INRIA Sophia Antipolis.

```
138.96.69.8 - - [03/Mar/2003 :18 :42 :14 +0100] "GET /axis/logoToile3.swf HTTP/1.1" -
-
138.96.69.8 - - [03/Mar/2003 :18 :48 :00 +0100] "GET /aid/personnel/ HTTP/1.1" - -
138.96.69.8 - - [03/Mar/2003 :19 :03 :14 +0100] "GET /axis/cbrtools/ HTTP/1.0" - -
138.96.69.7 - - [03/Mar/2003 :19 :04 :44 +0100] "GET /axis/cbrtools/manual/ HTTP/1.0"
- -
138.96.69.7 - - [03/Mar/2003 :19 :12 :45 +0100] "GET /axis/broadway/ HTTP/1.0" - -
```

FIG. 2.1 – Exemple de fichier access log

Deux types de traitements sont effectués sur les entrées du serveur log. Tout d'abord, le fichier access log est trié par adresse et par transaction. Ensuite, une étape d'élimination des données « non intéressantes » pour

l'analyse (phase de sélection) est réalisée. Au cours de la phase de tri et afin de rendre plus efficace le traitement de l'extraction de données, les URL et les clients sont codés sous forme d'entiers. Toutes les dates sont également traduites en temps relatif par rapport à la plus petite date du fichier.

Définition 4 Soit Log un ensemble d'entrées dans le fichier access log. Une entrée g, $g \in Log$, est un tuple $g =< ip_g, \{(l_1^g.URL, l_1^g.time), ..., (l_m^g.URL, l_m^g.time)\} >$ tel que pour $1 \leq k \leq m$, $l_k^g.URL$ représente l'objet demandé par le client g à la date $l_k^g.time$, et pour tout $1 \leq j < k$, $l_k^g.time > l_j^g.time$.

La figure 2.2 illustre un exemple de fichier obtenu après la phase de prétraitement. A chaque client correspond une suite de « dates » (événements) et la traduction de l'URL demandée par ce client à cette date.

Client	d1	d2	d3	d4	d5
1	10	30	40	20	30
2	10	30	60	20	50
3	10	70	30	20	30

FIG. 2.2 – Exemple de fichier résultat issu de la phase de prétraitement

L'objectif est alors de déterminer, grâce à une phase d'extraction, les séquences de ce jeu de données, qui peuvent être considérées comme fréquentes selon la définition 3. 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 2.2). Ce dernier résultat, une fois retraduit en termes d'URL, confirme la découverte d'un comportement commun à $minSup$ utilisateurs et fournit l'enchaînement des pages qui constituent ce comportement fréquent.

2.3 Travaux existants

Différentes techniques de fouille de données ont été appliquées au Web Usage Mining : les règles d'associations [CMS99, ZXH98, BGG+01], les motifs séquentiels [MPC00, SFW99, BGG+01, ZHH02, NM03, MTT03, Tan05], la classification de sessions [PPK+99, MDLN02, FSS00], d'utilisateurs [PPK+00, CHM+00] ou encore de pages [MDLN02]. Dans cette section, nous allons nous focaliser sur les principales techniques d'extraction de motifs séquentiels ont été appliquées aux logs d'accès Web.

Le principal intérêt d'utiliser ces techniques pour les données d'usage Web est la prise en compte de la temporalité.

Premières applications des motifs séquentiels pour analyser les usages du Web :

L'outil WUM (Web Utilisation Miner) proposé dans [SFW99] permet la découverte de patrons de navigation qui sont intéressants du point de vue statistique ou de leur structure. L'extraction de motifs séquentiels proposée par WUM repose sur la fréquence (support minimum) des motifs considérés. On peut aussi spécifier un autre critère subjectif pour les patrons de navigations comme par exemple le fait de passer par des pages avec certaines propriétés ou que la confiance soit élevée entre deux ou plusieurs pages du patron de navigation.

Dans [MPC00], les auteurs proposent la plateforme WebTool. L'extraction des motifs dans WebTool repose sur PSP, un algorithme développé par les auteurs, dont l'originalité est de proposer un arbre préfixé pour gérer à la fois les candidats et les fréquents.

Malheureusement, la dimensionnalité de ces données (tant en nombre d'items - pages - différents, qu'en nombre de séquences) pose des problèmes aux techniques d'extraction de motifs séquentiels. Plus précisément, à cause du nombre important d'items différents le nombre de résultats obtenus est très faible. La solution consisterait dans une baisse du support utilisé mais dans ce cas les algorithmes ne parviennent pas à finir et donc à fournir des résultats.

Une solution est de réduire le nombre d'items en utilisant une généralisation des URLs. Dans [FSS00] les auteurs utilisent une généralisation syntaxique des URLs avec un type diffèrent d'analyse (classification). Avant d'appliquer une classification par BIRCH [ZRL96], les rubriques syntaxiques de niveau supérieur à deux sont remplacées par leurs rubriques syntaxiques de niveau inférieur. Par exemple, au lieu de *http ://www-sop.inria.fr/axis/Publications/2005/all.html* ils utiliseront *http ://www-sop.inria.fr/axis/* ou bien *http ://www-sop.inria.fr/axis/Publications/*. Cependant cette généralisation syntaxique, même si elle est automatique, est naïve car elle s'appuie trop sur l'organisation qui a été donnée aux pages du site Web. Une mauvaise organisation va implicitement générer une mauvaise classification et donc des résultats de faible qualité. Dans [TT04], une généralisation basée sur des rubriques sémantiques est faite lors du pré-traitement de logs Web. Ces rubriques (ou catégories) sont données a priori par un expert du domaine relatif au site Web considéré. Cependant ceci se révéler une tâche

couteuse en temps aussi bien pour la définition que pour la mise à jour de telles catégories.

Enfin notons qu'il existe d'autres méthodes (comme [SA96]) pour extraire des motifs séquentiels en tenant compte d'une généralisation mais dans d'autres domaines que le Web.

Parmi les avantages que présentent les motifs séquentiels sur les règles d'association, dans le contexte du Web Usage Mining, nous soulignerons en particulier la prise en compte de la temporalité. Nous présentons ici trois ensembles d'études qui impliquent cette notion de temporalité pour l'analyse du comportement des utilisateurs d'un site Web.

Techniques d'extraction proches des motifs séquentiels :

Dans [HWV04] les auteurs proposent de considérer la temporalité qui caractérise les accès au site dans une méthode de classification des utilisateurs. Cette classification repose sur un algorithme d'alignement des séquences afin de mesurer leur distance. La principale contribution se situe alors dans la qualité des clusters proposés, qui sont comparés dans leurs expérimentations aux clusters obtenus avec des distances basées sur les itemsets uniquement.

Les auteurs de [ZHH02] voient les navigations des utilisateurs d'un site Web comme une chaîne de Markov. Ainsi ils proposent la construction d'un modèle de Markov pour une prédiction de liens tenant compte des navigation passées. L'article est consacré aux problèmes relatifs aux modèles de Markov et à la matrice de transition calculée pour chaque log. Les auteurs présentent un algorithme de compression de la matrice de transition afin de rendre la prédiction de liens plus efficace.

Qualité des résultats et prise en compte des caractéristiques du site :

Plus récemment, les travaux sur l'analyse des usages du Web se sont particulièrement penchés sur la qualité des résultats, leur pertinence et leur utilité. Cela est également le cas pour les travaux basés sur la prise en compte de la temporalité.

Dans [NM03] les auteurs montrent que les caractéristiques du site ont un impact sur la qualité des propositions qui seront faites aux utilisateurs selon que ces propositions sont calculées à partir d'itemsets fréquents ou de sequences fréquentes (ainsi que de motifs séquentiels). Principalement trois caractéristiques du site sont envisagées : la topologie, le degré de connectivité

et la longueur des navigations possibles. Les motifs sont d'abord extraits à partir de la première moitié du log. Pour chaque méthode d'extraction, les motifs sont utilisés pour évaluer la pertinence des prédictions qui en seraient déduites. Cette pertinence est évaluées sur la seconde moitié des navigations du log. Leur conclusion porte sur l'adéquation des itemsets pour les sites avec un très haut degré de connectivité et des chemins très courts, mais aussi l'adéquation des motifs séquentiels pour les sites avec des navigations longues (y compris les sites avec génération dynamique de pages).

2.4 Diviser pour découvrir

Cette section est basée sur l'article « Diviser pour Découvrir : une Méthode d'Analyse du Comportement de Tous les Utilisateurs d'un Site Web » (Florent Masseglia, Doru Tanasa et Brigitte Trousse) publié dans les actes de la conférence BDA 2003.

Dans cette section, nous présentons les motivations de notre travail, en termes d'intérêt des motifs visés et de difficultés engendrées, ainsi que le principe général de division que nous proposons.

2.4.1 Motivations

Considérons des sites web comme celui du siège de l'INRIA ou celui de l'unité de Sophia Antipolis. Ces sites peuvent être, en partie, représentés comme le suggère la figure 2.3. Il s'agit de sites riches, dont les thèmes peuvent varier de l'emploi à l'INRIA, au plan stratégique en passant par les pages des membres d'une unité (pages relatives aux activités de recherche, d'enseignement, etc., de chacun).

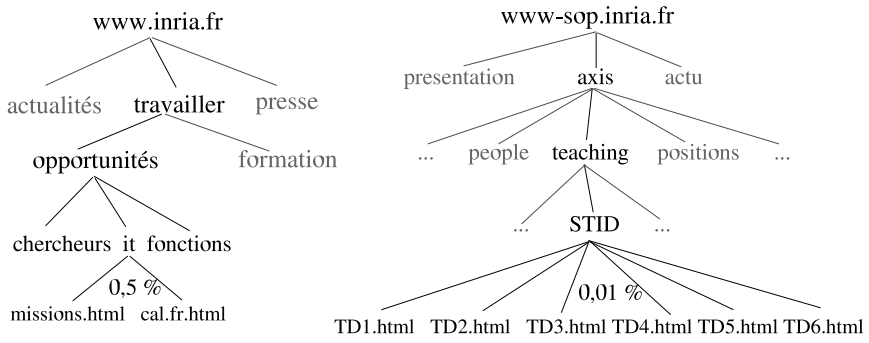

FIG. 2.3 – Une partie des sites de l'INRIA

Les leçons que l'on peut tirer d'une activité d'analyse du log correspondant à ces sites sont les suivantes :

- Généralement, les motifs séquentiels issus du fichier log d'un site de cette ampleur sont assez décevants. En effet, leur significativité est assez faible et leur évidence les rend peu utiles (par ex. « 0,1 % des utili-

sateurs sont passés par la page d'accueil puis la page du sommaire »).
- Les comportements intéressants sont contingentés à une partie très précise du log. Par exemple, sur la figure 2.3, la partie du log correspondant aux activités d'enseignement de D. Tanasa (STID) sera consultée par **0,01 %** des utilisateurs enregistrés dans le log. Les utilisateurs concernés par les opportunités de travail à l'INRIA, eux, représentent 0,5 % des accès sur le site.
- Si l'on veut extraire des motifs séquentiels intéressants sur ce log, il faut donc spécifier un support extrêmement bas.

Cette dernière réflexion nous permet de mettre en évidence les problèmes suivants. Dans notre cadre de travail, nous avons défini deux voies qui sont volontairement mises de côté. Tout d'abord, la recherche de motifs sous contraintes. Sans discuter l'efficacité de cette méthode qui a fait ses preuves, nous prenons cette position pour des arguments relatifs à la seule complétude des résultats. En spécifiant des contraintes, l'utilisateur oriente l'algorithme dans ses recherches. Nous considérons donc que cette technique ne permet pas la découverte de tous les motifs (qui sont par essence à découvrir, donc invisibles de l'utilisateur et de ses contraintes). La deuxième technique que nous avons écartée est celle de l'échantillonnage. En effet, compte tenu de la très faible représentativité des comportements que nous cherchons, la taille de l'échantillon à spécifier devrait approcher la taille du log. Dans ces conditions, une méthode d'échantillonnage verrait son principal atout remis en cause.

Imaginons maintenant que l'on baisse le support de manière trop importante. Deux conséquences devront alors être prises en compte :
- Le temps de réponse nécessaire au déroulement de l'extraction de motifs séquentiels avec ce support devient trop long (la plupart du temps, les résultats ne seront même jamais obtenus, en raison de la complexité du processus).
- Le nombre de fréquents générés par ce processus (dans le cas où il se termine) fait que les résultats seront difficiles à exploiter.

Pourtant, les comportements que nous voulons découvrir ont un support typiquement très faible. En effet ces comportements correspondent à des minorités, mais nous avons pour objectif de les découvrir car nous estimons qu'ils sont révélateurs et utiles. Par exemple, parmi ces comportements, nous pouvons compter les attaques pirates sur le site, ou encore la consultation (certainement par des étudiants) des pages d'exercices correspondant à une séance de travaux pratiques. Notre ambition est alors de mettre en évidence des comportements qui permettraient d'affirmer que :
- 0,04 % des utilisateurs ont une activité susceptible d'être une attaque

sur le site. Parmi eux, 90 % ont navigué selon une séquence typique d'une attaque pirate.
- 0,01 % des utilisateurs ont une activité relative à la partie « enseignement » de D. Tanasa. Parmi eux, 15 % ont consulté dans l'ordre les 6 pages de son cours sur le data mining.

Le support extrêmement faible de ces motifs est essentiellement dû à la grande diversité des comportements sur le log analysé et au grand nombre d'URL contenu dans ce site. Pour contourner les problèmes que nous venons de décrire et découvrir tout de même ces motifs, nous avons mis en place la méthode « diviser pour découvrir », décrite plus bas.

2.4.2 Principe général

Dans les grandes lignes, notre objectif est de découvrir des classes d'utilisateurs (regroupés en fonction de leur comportement sur le site) et d'analyser, ensuite, leurs navigations grâce à une extraction de motifs séquentiels. Notre méthode s'appuie donc sur deux phases. La première phase consiste à diviser le log en sous-logs, censés représenter des activités distinctes. La seconde phase consiste à analyser le comportement des utilisateurs enregistrés dans chacun de ces sous-logs. Le principe général de notre méthode est le suivant :

1. Extraire des motifs séquentiels sur le log d'origine.
2. Procéder à une classification sur ces motifs séquentiels. Cela nous permet d'obtenir une première classification sur le comportement des utilisateurs.
3. Diviser le log en fonction des classes ainsi obtenues. Chaque sous-log contient les sessions du log original qui correspondent à au moins un comportement de la classe ayant engendré ce sous-log. Un sous-log spécial est alors créé pour recueillir les sessions du log d'origine qui ne correspondent à aucune des classes obtenues dans l'étape précédente.
4. Pour chaque sous-log obtenu, réitérer l'ensemble de ce processus.

La figure 2.4 illustre cette façon de procéder. Dans le haut de cette figure, on peut observer l'obtention des motifs séquentiels sur le log d'origine, puis leur classification. En bas de cette figure est illustrée la division en sous-logs (SL_1 à SL_n) du log d'origine, en fonction des classes (C_1 à C_n) obtenues précédemment. On peut y constater la création du sous-log SL_{n+1} qui contient toutes les sessions n'ayant pas été reconnues comme appartenant aux comportements identifiés sur le log d'origine.

C'est sur ce dernier sous-log que va précisément reposer la qualité des résultats produits par notre approche. En effet, les premiers sous-logs obte-

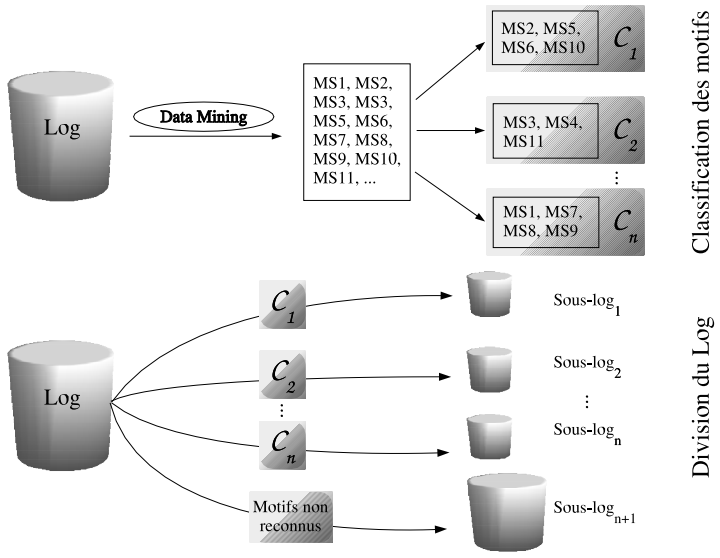

FIG. 2.4 – Principe de la méthode « diviser pour découvrir »

nus contiennent les catégories d'utilisateurs les plus représentés. Ils ont donc un intérêt certain, mais l'analyse la plus riche d'enseignements viendra de l'exploitation qui peut être faite du sous-log SL_{n+1} contenant les sessions « inclassables ». En considérant désormais ce sous-log comme un log « d'origine », et en réitérant le processus de division du log (tel qu'il est décrit par la figure 2.4), nous serons en mesure de découvrir des comportements dont la représentation est faible dans le log d'origine. La figure 2.5 illustre ce concept. Considérons les sessions inclassables de SL_{n+1}. Nous allons les placer dans le sous-log $SL_{n+1_{k+1}}$. Au final, c'est donc une arborescence dont la configuration est proche d'un peigne qui nous permettra de pousser ce processus de division jusqu'à l'obtention de comportements vraiment très minoritaires.

En analysant de cette façon les sessions contenues dans les logs de l'INRIA, nous avons pu constater l'existence de comportements avec une répartition semblable à celle illustrée par la figure 2.5. Tout d'abord, les comportements les plus représentés sur le log de l'INRIA sont ceux qui consistent à consulter, par exemple, les parties du site suivantes :

– **/travailler/opportunités/...**, qui regroupe les pages sur les opportunités d'emploi ;

- /**recherche**/**equipes**, qui est le portail sur toutes les équipes de recherche ;
- /**inria**/**organigramme**, l'organigramme de l'INRIA ;
- /**valorisation**/**logiciels**, qui est le portail des logiciels issus des projets de l'INRIA ;

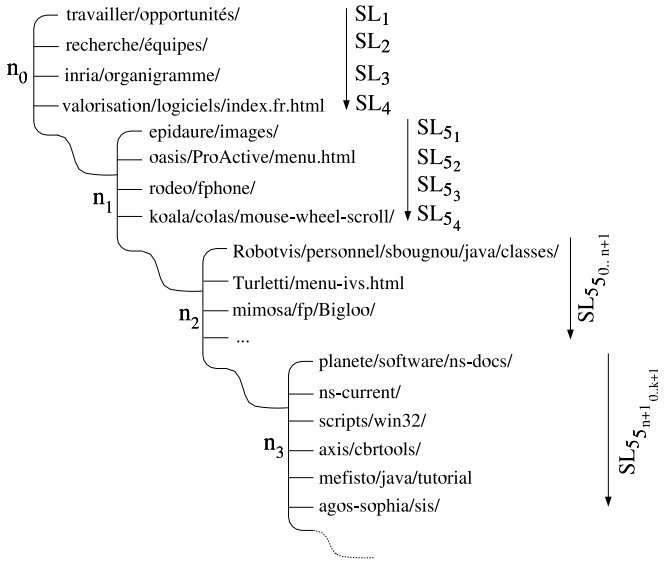

FIG. 2.5 – Division d'un log par étapes successives

Ensuite, en travaillant sur le sous-log SL_{n+1}, nous avons pu mettre en évidence des comportements relatifs à certaines équipes de recherche de Sophia Antipolis, comme *epidaure, oasis, rodeo* ou encore *koala*. En retravaillant sur les sessions non classées à cette étape (soit le sous-log $SL_{n+1_{k+1}}$), nous pouvons découvrir des comportements relatifs à certaines équipes (comme *mimosa* ou *robotvis*, mais aussi des pages du personnel (e.g. *turletti*). Enfin, en poursuivant notre processus de division, nous pourrons découvrir des comportements précis très cohérents, contingentés à des parties bien définies du site, et avec un support vraiment très faible pour le log original. Il s'agit par exemple de comportements relatifs à des équipes de recherche (*AxIS, Planete*), à l'association des œuvres sociales de l'INRIA (*Agos*), ou encore des tentatives d'intrusion (*/scripts/win32/...*).

Pour fournir des résultats aussi fiables, notre méthode dépend d'un facteur bien précis : la qualité de la division proposée pour un log donné. Or, cette division repose sur la façon dont les motifs séquentiels seront classés.

Nous avons étudié plusieurs méthodes de classification des motifs séquentiels. La plus efficace d'entre elles étant une méthode neuronale utilisant un résumé pour chaque motif séquentiel à classer basé sur une généralisation des séquences d'accès web. Pour plus de détails sur cette méthode, le lecteur est invité à consulter [BT02, Tan05].

2.4.3 Expérimentations

Les logs sur lesquels nous avons effectué nos expérimentations ont les caractéristiques décrites par les tableaux de la figure 2.6. Ils portent sur une période d'un mois (février 2003) pour le site du siège, deux mois (février et mars 2003) pour le site de Sophia Antipolis et représentent 2,1 Go et 3 Go. Les programmes d'extraction sont réalisés en C++ sur une machine de type PC équipée de processus pentium 2,1 Ghz et exploitée par un système Linux (2.4).

Caractéristique	www.inria.fr	www-sop.inria.fr
Nombre de lignes	11 637 62	15 158 076
Nombre de sessions	432 396	564 870
Nombre d'URL (filtrées)	68 732	82 372
Longueur moyenne des sessions	6,3	4,4
Nombre moyen d'URL dans les sessions	7,2	6,3

FIG. 2.6 – Caractéristiques des fichiers logs

Lors de nos expérimentations sur ces logs, nous avons pu mettre en évidence des comportements fréquents, dont la représentativité relative (support du comportement par rapport au nombre total de sessions du log) était de plus en plus faible. Cette baisse de la représentativité étant proportionnelle au nombre de divisions effectuées pour isoler ce comportement. Les tableaux de la figure 2.7 recensent quelques-uns des motifs découverts. Voici la description de ces comportements :

C1 : avec le préfixe commun **travailler/opportunites/** :
<(it.fr.html) (ita/missions.fr.html) (ita/concoursit.fr.html)
(ita/annales2001/index.fr.html)>. Ce comportement est relatif aux offres de postes d'ITA.

C2 : <(travailler/opportunites/chercheurs.fr.html)
(travailler/opportunites/chercheurs/concourscr2.fr.html)
(recherche/equipes/index.fr.html) (recherche/equipes/listes/index.fr.html)>.
Ce comportement est relatif aux offres de postes de chercheurs à l'INRIA.

www.inria.fr

Id	P	Sessions	S1	S2	T1	T2	R1	R2
C1	1	28740	10 %	0,6 %	34	23	27	70
C2	2	4473	28 %	0,28 %	58	1364	188	23035
C3	3	280	85 %	0,04 %	73	?	14	?
C4	4	198	13 %	0,006 %	97	?	6	?

www-sop.inria.fr

Id	P	Sessions	S1	S2	T1	T2	R1	R2
C5	1	19686	10 %	0,34 %	24	24	1	35
C6	2	3252	4 %	0,02 %	51	?	138	?
C7	2	1551	29 %	0,08 %	74	16681	20	2482
C8	3	381	23 %	0,01 %	99	?	6	?

Description des paramètres

Id	Identifiant du comportement
P	Niveau de profondeur (nombre de divisions nécessaires pour l'obtenir)
Sessions	Nombre de sessions du sous-log dont ce comportement est extrait
S1	Support relatif du comportement (support sur le sous-log)
S2	Support absolu du comportement (support sur le log d'origine)
T1	Temps (s) relatif nécessaire pour obtenir ce comportement (sur le sous-log)
T2	Temps (s) absolu pour obtenir ce comportement avec le support S2 (sur le log d'origine)
R1	Taille relative du résultat (nombre de comportements sur le sous-log)
R2	Taille absolue du résultat avec le support S2 (nb de comportement sur le log d'origine)

FIG. 2.7 – Caractéristiques des motifs découverts

Les utilisateurs consultent la page des offres, celle des concours puis celles décrivant les équipes.

C3 : <(scripts/root.exe) (c/winnt/system32/cmd.exe)
(..%255c../..%255c../winnt/system32/ cmd.exe)
(..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c../winnt/system32/cmd.exe)
(winnt/system32/cmd.exe) (winnt/system32/cmd.exe) (winnt/system32/cmd.exe)>.
Ce comportement est typique d'une attaque pirate. Après l'avoir isolé, nous avons consulté le responsable de la sécurité du réseau de l'unité de Sophia Antipolis, qui nous a confirmé qu'un tel enchaînement d'appels aux scripts ne pouvait être qu'une attaque y recherchant une faille. Généralement, ces

attaques sont préprogrammées et les pirates utilisent les mêmes programmes d'attaques, ce qui confère à ce comportement un support relatif très élevé (plus de 80 %).

C4 : avec le préfixe commun **rapportsactivite/RA95/omega/** : <(node10.html) (node11.html) (node12.html) (node13.html)>. Ce comportement reflète l'activité des utilisateurs qui se sont intéressés au rapport d'activité du projet *omega*. Cependant, cette activité porte sur le rapport de 1995. Cela pourrait s'expliquer par le fait que les moteurs de recherche disponibles sur internet (extérieurs à l'INRIA) renvoient sur les rapports d'*omega* des années 1995, 1997 et 1998 quand l'objet de la recherche correspond au thème de ce projet. Cela illustre nos propos, en introduction, disant que les moteurs de recherche extérieurs sont un facteur incontournable.

C5 : <(koala/colas/mouse-wheel-scroll) (koala/colas/mouse-wheel-scroll)>. Parmi les comportements extraits, tous ne sont pas d'un intérêt flagrant. Celui-là montre simplement une répétition de l'URL pointant sur un logiciel développé par un membre de l'unité de Sophia Antipolis. Ce logiciel étant fortement demandé, il ressort parmi les premiers comportements.

C6 : avec le préfixe commun **robotvis/personnel/zzhang/Publis/Tutorial-Estim/** : <(Main.html) (node3.html) (node4.html) (node5.html) (node6.html) (node7.html)>. Cet exemple, comme le suivant, reflète le comportement d'utilisateurs venus consulter des pages de cours ou de tutoriaux réalisés par des membres du personnel de l'unité.

C7 : avec le préfixe **mascotte/personnel/Sebastien.Choplin/cours/iut-infocom/ excel/** : <(exercices.html) (exo1.xls) (exo2.xls) (exo3.xls) (exo4.xls) (exo5.xls)>.

C8 : avec le préfixe **epidaure/Demonstrations/foie3d/** : <(endo4.html) (endo5.html) (endo6.html) (endo8.html) (endo9.html) (endo10.html) (endo11.html) (endo12.html)>.

2.5 Usages du Web généralisés

Cette section est basée sur l'article « GWUM : une généralisation des pages Web guidée par les usages. » (Doru Tanasa, Florent Masseglia et Brigitte Trousse) publié dans les actes de la conférence Inforsid 2006

Comme nous l'avons indiqué dans l'introduction, la généralisation des items est un facteur clé lors de l'extraction de motifs séquentiels. Pour comprendre l'enjeu de nos travaux, nous proposons l'exemple suivant.

Client	Date1	Date2	Date3
C1	accueil_DT	publications_DT	accueil_Inria
C2	accueil_SC	publications_SC	logiciels_AxIS
C3	accueil_DT	publications_AxIS	publications_DT
C4	accueil_AxIS	accueil_SC	publications_SC

TAB. 2.1 – Accès au site regroupés par client

Considérons les enregistrements du log de l'Inria Sophia-Antipolis reportés dans le tableau 2.1. On peut y lire que le client 1 à la date 1 a fait une requête sur l'URL "accueil_DT" qui est la page d'accueil de Doru Tanasa, puis à la date 2 une requête sur la page des publications de Doru Tanasa et enfin une requête sur la page d'accueil de l'Inria. De la même manière, le client 2 a fait une requête sur la page d'accueil de Sergiu Chelcea, et ainsi de suite...
Avec une analyse basée sur les motifs séquentiels et un support de 100%, aucun motif ne sera trouvé dans ce log (aucun item n'est partagé par 100% des enregistrements). Pour trouver un comportement fréquent, il faudra descendre le support jusqu'à un seuil de 50%, ce qui permet d'extraire les comportement suivants :

1. $<$ (accueil_DT) (publications_DT) $>$ (vérifié pour les clients C1 et C3)
2. $<$ (accueil_SC) (publications_SC) $>$ (vérifié pour les clients C2 et C4)

Malheureusement, le fait de baisser le support :

1. Est un facteur de ralentissement, voir de bloquage, du processus d'extraction.
2. Retourne généralement des résultats difficiles à interpréter car nombreux et similaires (ou redondants).

Considérons maintenant que nous soyons en mesure de classer les URLs de ce log dans différentes catégories. Par exemple la catégorie "Pub" contiendrait les pages relatives aux publications de chercheurs (dans notre cas : "publications_DT" et "publications_SC"). La catégorie "Mining" contiendrait les pages relatives au data mining (dans notre cas les pages d'accueil de Doru Tanasa et Sergiu Chelcea qui ont fait leur travail de thèse sur ce thème). Avec de telles informations, nous serions en mesure d'extraire un motif avec un support de 100% qui serait : $<$ (Mining) (Pub) $>$. L'interprétation de ce motif est que 100% des utilisateurs consultent une page relative au data mining puis une page relative à des publications de chercheurs. On peut en effet vérifier ce comportement sur les enregistrements du tableau 2.1.

2.5.1 Méthode

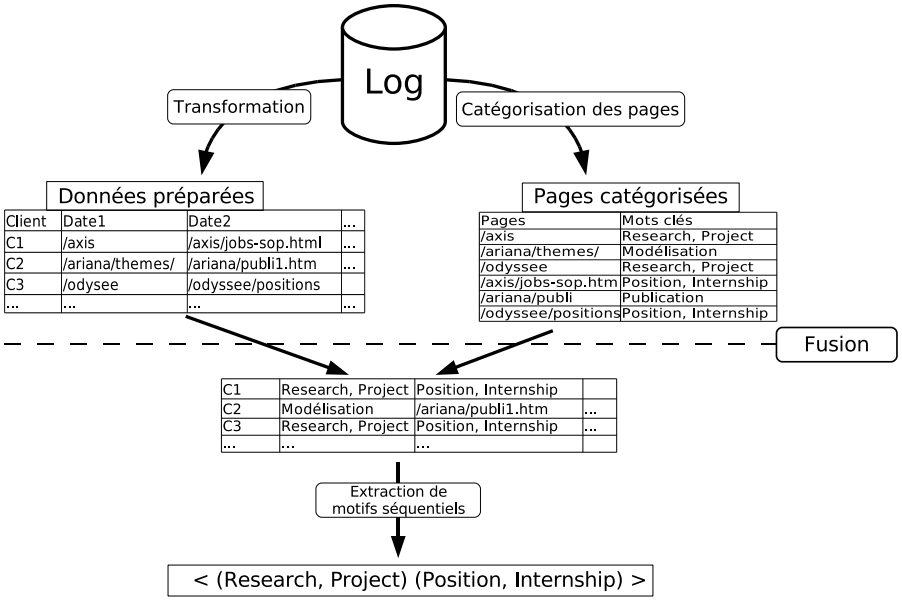

FIG. 2.8 – Méthode de catégorisation et d'analyse des usages

Notre méthode est illustrée par la figure 2.8. A partir du fichier log, notre objectif est d'obtenir des informations sur les pages Web, afin de les catégoriser. Une ligne d'enregistrement dans le fichier log se présente de la

manière suivante :

```
111.222.111.222 - - [01/Oct/2005:10:45:05 0200]
"GET /axis/Publications/ HTTP/1.1" 200 3754
"/axis" "Mozilla/4.0"
```

Dans cet enregistrement, on peut trouver les informations suivantes (entre autres) :

- la machine d'IP `111.222.111.222`
- à la date du `[01/Oct/2005:10:45:05]`
- a demandé la page `/axis/Publications/`
- `3754` octets ont été transférés
- la page précédente était `/axis` (information connue sous le nom de *referrer*).
- et le navigateur `Mozilla/4.0`

Ces informations constituent une première source d'information sur la page (façon dont elle est accédée, navigateur utilisé, etc.) qui nous renseignent sur les usages qui en sont fait. Un premier ensemble de catégorie est alors envisageable à partir de ces informations. Il peut s'agir par exemple de faire une catégorisation selon le pays d'origine des IP qui accèdent aux différentes pages. Cela permettrait de classer les pages selon qu'elles sont plus accédées à partir de la France, des USA, du Japon, etc. Il peut également s'agir de travailler sur la taille de la page (par exemple une page pourrait être catégorisée comme étant de "taille moyenne").

Une deuxième catégorisation peut être réalisée à partir des informations concernant la page elle-même. Par exemple, une analyse du contenu de la page d'accueil du projet AxIS peut montrer que ses mots clés sont "Welcome", "Project" et "Research". Encore une fois, l'application d'une méthode de classification sur l'ensemble des pages Web accédées dans le log à l'aide de ces mots clés permettra d'obtenir une catégorisation des pages. Dans le cas de la page d'accueil du projet AxIS la figure 2.8 montre que cette page serait catégorisée comme "Research, Project".

Enfin, une fois ces catégories extraites, notre objectif est de les exploiter lors de l'analyse des usages. Par exemple, avec les motifs séquentiels, il s'agirait de préparer les données dans le format approprié à l'extraction de motifs séquentiels avec en tant qu'item une catégorie de l'URL plutôt que l'URL elle-même. Dans le cas de la figure 2.8 la page d'accueil du projet AxIS aurait pour item "/axis" (*e.g.* l'item du client 1 à la date 1). Si l'on veut exploiter la catégorisation 1, alors l'item "/axis" sera remplacé par sa catégorie ("Research, Project"). L'extraction de motifs séquentiels sur de telles données peut alors aboutir à des motifs comme < (Research, Project) (Position, Internship) > qui serait interprété comme « x% des utilisateurs consultent

une page du type *Projet de recherche* suivie d'une page relative à des *offres d'emploi ou de stage* ».

2.5.2 Extraction d'informations en vue d'une classification automatique des pages

Pour instancier notre principe de généralisation des pages, nous proposons une extraction d'informations relatives à l'accès à ces pages (utilisation du champ '*referer*'). Nous avons utilisé le champ referer d'une ligne HTTP quand celui-ci contenait une requête issue d'un moteur de recherche. Nous en avons extrait les mots clés. Les mots clés utilisés dans une requête sont ensuite passés dans l'outil TreeTagger développé à l'Institut de Linguistique Computationnelle de l'Université de Stuttgart [Sch94]. TreeTagger marque les mots d'un texte avec des annotations grammaticales (nom, verbe, article, etc.) et transforme les mots en leur racine syntaxique (lemmatisation).

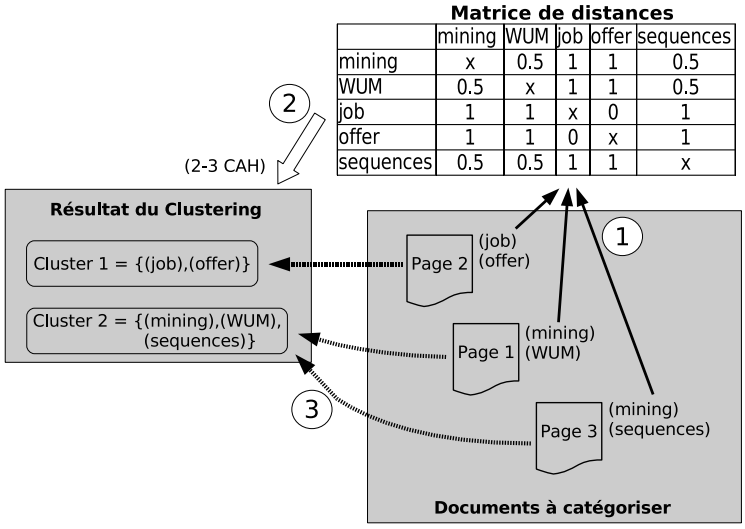

FIG. 2.9 – Catégorisation des pages Web, basée sur le referrer

L'objectif est d'obtenir une catégorisation des pages qui soit guidée par les usages. Nous considérons en effet que les mots clés qui sont fréquemment employés dans un moteur de recherche pour accéder à une page, peuvent être utilisés pour caractériser cette page. Ces mots clés sont accessibles dans le champs *referrer* d'un enregistrement du log. Dans une seconde phase,

nous utilisons l'ensemble de ces caractéristiques pour faire des rapprochement entre les pages. Par exemple, si la page Web de Doru Tanasa est caractérisée par les mots clés "Data Mining" et "WUM", et que la page Web de Florent Masseglia est caractérisée par les mots clés "Data Mining" et "Sequential Patterns" alors il serait logique de créer une catégorie de pages "Data Mining" qui contiendrait ces deux pages Web. Cela se traduirait par le fait que, selon les utilisateurs qui ont accédés à ces pages suite à une requête sur un moteur de recherche, ces pages sont relatives au "Data Mining".

La procédure que nous avons mise en place pour catégoriser les pages est décrite par la figure 2.9. Notre but est d'obtenir une classification des URLs. L'idéal aurait été de construire une matrice de dissimilaritudes pour les URLs et de procéder ensuite à une classification. Malheureusement, lors de nos expérimentations, la forte dimensionnalité des données ne nous a pas permis de procéder de la sorte. Nous avons, en effet, obtenu 62 721 URLs, décrites par un total de 35 367 mots clés. La matrice aurait atteint la taille de 62 721 $\times$ 35 367 (soit $2,2 \cdot 10^9$). Nous avons donc opté pour une première phase de classification sur les mots clés (étape 2 de la figure 2.9). Pour cela, nous proposons de construire la matrice de distances entre les mots clés (étape 1 de la figure 2.9). La distance entre deux mots clés a et b (*i.e.* $Dist(a,b)$) étant donnée par la formule suivante (indice de dissimilarité de Jaccard) : $Dist(a,b) = 1 - \frac{P_{ab}}{P_a + P_b - P_{ab}}$, ou P_x est le nombre d'URLs distinctes ayant été accédées par le mot clé x et P_{xy} est le nombre d'URLs distinctes ayant été accédées par le mot clé x et par le mot clé y. Cette mesure varie de 0 (les mots sont très similaires) à 1 (les mots sont très distants). Une fois cette matrice construite, nous avons procédé à la classification des mots clés, grâce à la 2-3CAH [CBT04, Che07] (étape 2, dans la figure 2.9). Nous avons alors obtenu l'ensemble C des clusters de mots clés. La dernière étape a consisté à affecter les URLs aux différents cluster. Pour cela nous avons traité l'ensemble U des URLs de la façon suivante (qui correspond à l'étape 3 de la figure 2.9) :

$\forall u \in U, \forall c \in C, \forall d \in C,\ si\ \frac{mots(u,c)}{|c|} \geq \frac{mots(u,d)}{|d|}\ alors\ c \leftarrow u$

Avec $mots(u,c)$ le nombre de mots clés partagés par u et par c. Cette procédure permet d'affecter les pages dans les clusters qui contiennent les mots leur correspondant le mieux. Les mots clés de ce cluster permettront de le décrire. Lors de nos expérimentations, nous avons par exemple obtenu les clusters de pages suivants :

- Cluster *log,fouiller* :
  - http ://www-sop.inria.fr/axis/personnel/Florent.Masseglia/
  - http ://www-sop.inria.fr/axis/personnel/Doru.Tanasa/papers/
  - http ://www-sop.inria.fr/axis/fdc-egc04

  - ...
- Cluster *Internet,audio* :
  - http ://www-sop.inria.fr/rodeo/fphone/
  - http ://www-sop.inria.fr/rodeo/fphone/changes.html
  - http ://www-sop.inria.fr/interne/accueil/audioconference.shtml
  - ...

Le premier cluster contient des pages relatives à la fouille de fichiers access log. Le second cluster contient des pages relatives au communications audio via Internet. On y trouve des pages du projet Rodeo (dont c'est l'un des thèmes majeurs) mais aussi la page du service des audio-conférences de l'unité Inria de Sophia-Antipolis.

2.5.3 Expérimentation

GWUM a été implémenté en Java et s'appuie sur PSP de [MCP98a]. L'expérimentation a été menée sur un Pentium avec des données issues des logs HTTP du site www-sop.inria.fr du mois d'octobre 2005. Ces données contiennent 258 061 navigations et 845 208 requêtes. La longueur moyenne des navigations est de 3.3 pages et il y a 114 238 navigations de longueur supérieure à 1.

Pré-traitement des données et catégorisation

Sur les 845 208 requêtes HTTP du log, 164 000 requêtes (presque 20%) ont été accédées par une requête dans un moteur de recherche. A partir de ces requêtes, sur les 35 367 mots différents extraits, seuls les racines des mots reconnus par l'outil TreeTagger ont été gardés en vue d'une catégorisation.

	Nombre	Urls différentes
Requêtes	845 208	62 721
Requêtes avec referer non vide	593 564	53 573
Requêtes avec referer *moteur de recherche*	164 000	17 671

TAB. 2.2 – Données du log considéré

Nombre de motifs séquentiels

L'objectif de cette section est de montrer l'intérêt de l'approche par caractérisation des URLs dans un processus de Web Usage Mining en terme de support minimum et de nombre de motifs extraits. En effet, notre but est de montrer que le support minimum peut être singulièrement augmenté avec

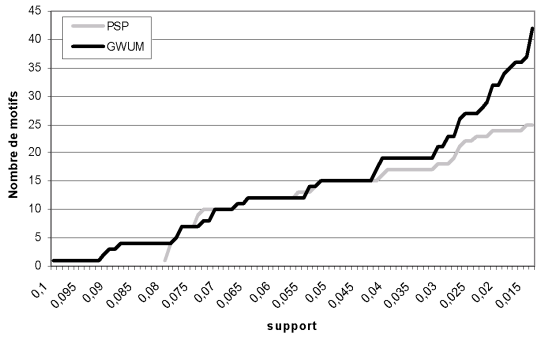

FIG. 2.10 – Nombre de motifs bruts extraits à différents supports avec (GWUM) et sans (PSP) généralisation des URLs

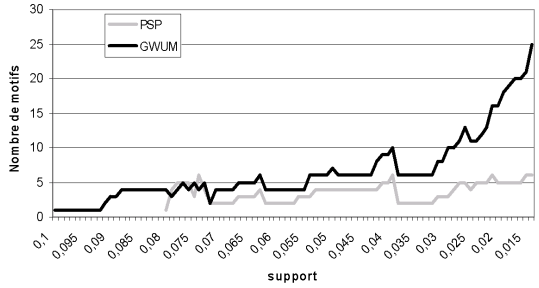

FIG. 2.11 – Nombre de motifs de taille maximale extraits à différents supports

une généralisation des URLs. La figure 2.10 montre en effet le nombre de motifs extraits à différents supports pour les données décrites dans la section 2.5.3. Il s'agit du nombre de motifs total (contrairement aux motifs de taille maximale). On peut y observer, par exemple, que pour un support de départ fixé à 10%, le nombre de motifs généralisés extraits est de 2 alors qu'aucun motif basé sur les URLs uniquement ne peut être trouvé. Il faut atteindre un support minimum de 7,5% avant qu'apparaissent deux motifs basés sur les URLs uniquement. Pour ce même support, on trouve 4 motifs généralisés.
Dans la figure 2.11 nous reportons le nombre de motifs de taille maximale. On peut y observer qu'à partir d'un support minimum d'environ 3%, ce nombre est nettement plus élevé pour les motifs séquentiels généralisés. Par exemple pour un support de 1%, le nombre de motifs trouvés par GWUM est de 25, contre 6 pour PSP. La différence du nombre de motifs bruts entre les

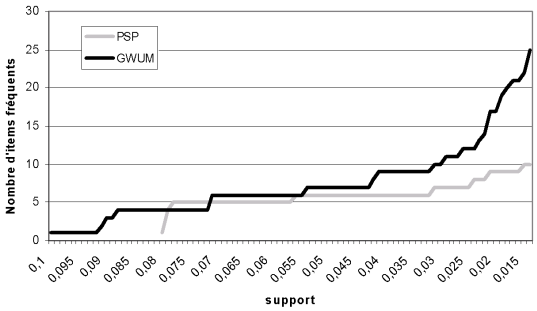

FIG. 2.12 – Nombre d'items extraits à différents supports

deux méthodes est moins importante que la différence du nombre de motifs de taille maximale. Cela est particulièrement dû au fait que pour un support x, GWUM va extraire des items fréquents qui ne seront pas découverts par PSP. Le nombre de motifs de taille maximale est donc très proche du nombre de motifs bruts, dans la mesure où de nombreux motifs sont constitués seulement d'un item qui n'est inclus dans aucun autre motif (en d'autres termes, cet item vient d'apparaître pour le support x). Cela peut être utile à l'utilisateur final pour faire un premier tri dans ses résultats et continuer à faire une fouille plus précise par la suite. Par exemple, l'apparition de l'item "publications_Tanasa" pour le support 0,5 % peut permettre une seconde phase de fouille sous contrainte. Cette seconde phase consisterait alors à n'explorer que les navigations qui contiennent l'item "publications_Tanasa". Enfin, dans la figure 2.12 nous reportons le nombre d'items extraits avec et sans généralisation des URLs.

2.6 Discussion

Ce chapitre a présenté deux approches destinées à répondre aux problèmes posés par l'extraction de motifs à très faibles supports.

2.6.1 Des motifs aux supports faibles...

Premièrement, nous avons proposé une méthode divisive basée sur un principe d'exploration récursive des données. Nous avons pu souligner la capacité de notre approche à extraire des comportements relatifs à des minorités d'internautes. Ces comportements ont des caractéristiques typiques des enjeux du data mining comme leur cohérence avec la communauté à laquelle ils sont associés mais aussi leur grande significativité. Notre première approche présente des caractéristiques innovantes en matière d'analyse des usages :

1. **Des comportements significatifs avec un support vraiment très faible.** La liste des comportements ainsi découverts couvre plus de 50 objectifs de navigation distincts sur le site du siège, et plus de 100 sur celui de l'unité de Sophia Antipolis. Nous avons reporté 8 objectifs différents, qui vont des offres de postes aux consultations de pages de cours, en passant par les activités de recherche et les tentatives de piratage. Les comportements reportés dans nos expérimentations ont donc pour but d'illustrer le type de comportements obtenus mais aussi le succès de notre méthode pour découvrir les sortes de « niches » décrites dans ce chapitre. À savoir des comportements homogènes pour une minorité d'utilisateurs, mais que l'on ne pourrait pas découvrir sans tenir compte de leur représentativité très faible sur le log global.
2. **Une méthode performante.** Pour confirmer nos propos, nous avons reporté dans la table de la figure 2.7 d'un côté le temps $T1$ nécessaire pour l'extraction du motif avec notre méthode et un support $S1$ (temps cumulé de l'extraction de motifs séquentiels à chaque étape de la division), et d'un autre côté le temps $T2$ nécessaire pour obtenir ce même motif avec une méthode classique, un support $S2$ correspondant à la représentativité de ce motif. Par exemple, si $S1$ est de 10 % pour un sous-log contenant 100 sessions et que le log original contient 100 000 sessions, alors $S2$ vaudra 0,1 %. Nous avons comparé les temps de réponse avec $S1$ sur le sous-log et $S2$ sur le log d'origine. Très souvent, le temps $T2$ nécessaire pour obtenir les résultats avec le support $S2$ est tel que nous n'avons pas pu obtenir les résultats. Le ' ?' traduit un échec de la méthode d'extraction classique dû à la faiblesse du support

et donc à la complexité du processus d'extraction. Cela traduit également l'incapacité d'une méthode d'extraction de motifs séquentiels classique à obtenir ces motifs.

3. **Des résultats plus faciles à exploiter.** De plus, avec un support $S2$ si faible les résultats peuvent atteindre une taille si grande qu'ils en deviendraient difficiles à exploiter. Grâce à notre méthode de division, les résultats sont classés au fil de leur découverte, en fonction de l'objectif de navigation du sous-log qui leur correspond.

2.6.2 ... et des supports augmentés par la généralisation

Deuxièmement, nous avons présenté une méthode d'analyse des usages basée sur une généralisation des URLs via une catégorisation des pages que ces URLs référencent. Les informations extraites pour ces pages en vue d'une catégorisation concernent l'accès aux pages par les utilisateurs (informations obtenues par une analyse du champ referer dans les logs HTTP). Les expérimentations que nous avons menées ont permis de souligner le gain obtenu par une telle approche découvrant des motifs séquentiels fréquents avec un support plus élévé. De plus l'interprétation des motifs est facilitée par les labels donnés à chaque catégorie de pages. Pour illustrer le potentiel de notre méthode en terme d'interprétabilité des résultats, mais aussi son impact sur le support minimum, voici deux exemples choisis parmi nos résultats dans cette section.

Le premier motif se présente sous la forme suivante :
< **(c,code,programme,source) (software,free)** >.

Ce motif traduit un comportement d'utilisateurs ayant consulté une page relative au code source de programmes écrits en C. Son support est de 0.2%. Nous avons trouvé plusieurs déclinaisons possibles de ce comportement dans le log brut (sans catégorisation des URLs) :

1. < (www-sop.inria.fr/mimosa/fp/Bigloo/)
   (www-sop.inria.fr/mimosa/fp/Bigloo/doc/bigloo.html) >. En effet, la première page de ce motif appartient à la classe *(c, code, programme, source)* et la deuxième appartient à la classe *(software, free)*. De plus, nous avons pu constater, à la lecture de ces pages sur le Web, que cette appartenance était justifiée par les thèmes abordés par ces pages. Le support de ce comportement est de 0.13%.
2. < (www-sop.inria.fr/oasis/ProActive/home.html)
   (www-sop.inria.fr/oasis/proactive/(...)/C3DRenderingEngine.java.html)>

avec un support de $2.6 \cdot 10^{-5}$.

3. Et 84 autres motifs similaires au précédent avec un support proche de zéro, mais dont l'accumulation dans les classes *c,code,programme,source* et *software,free* fait monter le support du motif généralisé.

Le second motif est le suivant :
< (**projet,informatique,rechercher**) (**emploi,offrir,offre**) >.

Ce motif traduit un comportement d'utilisateurs ayant consulté une page liée à la notion de projet de recherche suivie d'une page relative à une (des) offre(s) d'emploi(s). Ce comportement (de support 0.65%), se décline de plusieurs façons dans le log brut, dont par exemple :

1. < (www-sop.inria.fr/) (www-sop.inria.fr/actu/actu_emploi_actuel_fr.shtml) > avec un support de 0.33%

2. < (www-sop.inria.fr/) (www-sop.inria.fr/odyssee/positions/index.fr.html) > avec un support de $5.25 \cdot 10^{-5}$.

3. < (www-sop.inria.fr/planete/index-fr.html) (www-sop.inria.fr/act_recherche/stages_theses_fr.shtml) > avec un support de $8.75 \cdot 10^{-6}$.

4. < (www-sop.inria.fr/odyssee/presentation/index.fr.html) (www-sop.inria.fr/odyssee/positions/index.fr.html) > avec un support de 0.018%.

5. Et 92 autres comportement relatifs aux offres d'emploi à l'Inria.

Pour ce deuxième comportement généralisé, on peut observer que toutes les combinaisons possibles sont présentes. Il peut s'agir de naviguer de la page d'accueil de l'Inria (qui est dans la catégorie *(projet, informatique, rechercher)*) vers une offre d'emploi sur le site de l'unité de recherche (UR) ou sur un site de projet. Il peut également s'agir de naviguer d'un site de projet de recherche vers la page d'emplois de ce projet ou la page d'emplois de l'UR. De manière générale, toutes ces déclinaisons contribuent à l'augmentation du support de leur motif généralisé et aucune d'elles ne serait trouvée avec le support de leur motif généralisé, ce qui contribue à justifier le bien-fondé de notre approche.

2.6.3 Ce que nous enseignent ces supports très faibles

Comme je l'indiquais en introduction, le fait que les motifs extraits des systèmes de grande écoute[1] ont des support extrêmement faibles n'est qu'un symptôme. Ce symptôme nous permet cependant d'ouvrir des pistes de recherche pour l'expliquer et pour aller plus loin dans la recherche de solutions. Pour cela, il faut identifier des causes supplémentaires à cette faiblesse du support. Nous avons vu que la grande diversité des objets et des usages était une première cause principale. Dans la suite de mes travaux j'ai voulu montrer l'existence d'autres causes, et y apporter des solutions. La première de ces causes concerne le lien entre les motifs d'usage et l'actualité des objets qu'ils contiennent. Les systèmes et les comportements sont évolutifs et les motifs peuvent donc émerger, grandir, diminuer et disparaitre. Imaginons un motif qui apparaît en janvier et dure 2 jours avec une grande intensité. Une analyse des données de toute une année risque de ne pas le détecter car son support est élevé sur deux jours mais pas sur l'année entière. C'est une des causes du support faible. Le chapitre suivant est consacré à la détection de périodes sur lesquelles des motifs sont fréquents.

[1] *i.e.* des systèmes dont les usages sont très nombreux et très variés.

Chapitre 3

Extraction de schémas exprimant l'évolution : de nouveaux critères pour la fouille de données

Les techniques de fouille de données existantes pour l'analyse des usages sont actuellement basées sur un découpage des données arbitraire (*e.g.* "un log par mois") ou guidé par des résultats supposés (*e.g.* "quels sont les comportements des clients pour la période des achats de Noël ? "). Ces approches souffrent des deux problèmes suivants. D'une part, elles dépendent de cette organisation arbitraire des données au cours du temps. D'autre part elles ne peuvent pas extraire automatiquement des "pics saisonniers" dans les données stockées. Dans ce chapitre j'expose deux méthodes répondant à ces problèmes.

Premièrement, la section 3.4 propose d'exploiter les données (et plus particulièrement les comportements fréquents) pour découvrir de manière automatique des périodes "denses" de comportements. La méthode proposée extrait également, parmi l'ensemble des combinaisons possibles, les motifs séquentiels fréquents associés à ces périodes. Une période sera considérée comme "dense" si elle contient au moins un motif séquentiel fréquent **pour l'ensemble des utilisateurs qui étaient connectés sur le site à cette période**. Les expérimentations menées montrent l'efficacité et la pertinence de notre approche pour obtenir les motifs séquentiels fréquents et les périodes denses associées.

Deuxièmement, la section 3.5 aborde le problème de l'extraction d'itemsets associés à leurs périodes de fréquences. L'extraction d'itemsets fréquents est un sujet majeur de l'ECD et son but est de découvrir des corrélations entre les enregistrements d'un ensemble de données. Cependant, le support est calculé en fonction de la taille de la base dans son intégralité. Dans la section 3.5, nous montrons qu'il est possible de prendre en compte des périodes difficiles à déceler dans l'organisation des données et qui contiennent des itemsets compacts, qui représentent un comportement cohérent sur une période spécifique et nous présentons l'algorithme DeICo qui permet leur découverte. Il s'agit d'extraire des itemsets en optimisant deux critères. Le premier critère est le support de l'itemset sur une fenêtre de temps spécifique et le second critère est la taille de cette fenêtre. L'objectif est d'extraire **des motifs qui ont une fréquence supérieure au seuil minimum donné par l'utilisateur et la fenêtre de temps sur laquelle ce motif est fréquent**. Nous ajoutons alors deux contraintes sur la taille de la fenêtre qui doit être optimale. En ce sens :

- La fenêtre ne doit pas être contenue dans une autre fenêtre sur laquelle le motif est fréquent avec une couverture plus grande (car cela signifierait que des transactions utiles au support sont exclues de la fenêtre).
- La fenêtre ne doit pas contenir une autre fenêtre plus petite sur laquelle le motif est fréquent avec une couverture identique (car cela signifierait que des transactions inutiles au support sont conservées dans la fenêtre).

3.1 Activités

Mes travaux relatifs à ce thème ont donné lieu a des collaborations avec le LIRMM (pour l'algorithme Perio décrit dans la section 3.4 et pour le co-encadrement de Céline Fiot évoqué en section 3.6.1).

3.1.1 Encadrement

- **Post-doc** : Co-encadrement du post-doc de Céline Fiot. J'ai co-encadré Céline dans une collaboration avec le LIRMM sur le sujet de l'extraction de motifs séquentiels co-évolutifs. Les travaux correspondants sont décrits dans la section 3.6. Co-encadrantes : Anne Laurent et Maguelonne Teisseire. Taux de participation à l'encadrement de Céline Fiot : 50%.

– **Master** : J'ai dirigé le stage de Master Reherche 2ème année de Bashar Saleh en 2007. Le stage de Bashar Saleh portait sur l'extraction d'itemsets compacts. Les travaux correspondants sont décrits dans la section 3.5. Titre du rapport de stage de master : « Optimizing The Division Of Data For Knowledge Discovery » (Juin 2007).

3.1.2 Publications

Les travaux décrits dans ce chapitre ont donné lieu à des publications dans une revue internationale, dans une conférence internationale, dans des conférences nationales et dans un atelier international :

– Florent Masseglia and Pascal Poncelet and Maguelonne Teisseire and Alice Marascu. « Web Usage Mining : Extracting Unexpected Periods from Web Logs ». In Data Mining and Knowledge Discovery (DMKD) Journal. Springer Netherlands (Ed). 16(1) : 39-65 (2008).
– B. Saleh and F. Masseglia. « Time Aware Mining of Itemsets ». In Proceedings of the Fifteenth International Symposium on Temporal Representation and Reasoning (TIME 08), Montréal, Jun 16-18. 2008.
– B. Saleh and F. Masseglia. Extraction de motifs séquentiels compacts. Extraction et Gestion des Connaissances (EGC 08), Sophia-Antipolis, Jan. 29-Feb 1st 2008.
– Florent Masseglia, Pascal Poncelet, Maguelonne Teisseire and Alice Marascu. « Usage Mining : extraction de périodes denses à partir des logs Web ». Extraction et Gestion des Connaissances (EGC'06), Lille, January 2006.
– F. Masseglia, P. Poncelet, M. Teisseire and A. Marascu. « Web Usage Mining : Extracting Unexpected Periods from Web Logs ». In IEEE 2nd Workshop on Temporal Data Mining (TDM'05). Held in conjunction with ICDM'05, Houston, USA, November 27, 2005.

3.2 Definitions

La définition 5 reprend le concept d'itemset fréquent de [AIS93]. Nous y avons ajouté la notion d'estampille (donc une transaction peut couvrir plusieurs dates).

Définition 5 *Soit* $\mathcal{I} = i_1, i_2, ..., i_n$ *un ensemble d'items. Soit* $X = \{i_1, i_2, ..., i_k\}/k \leq n$ *et* $\forall j \in [1..k]$ $i_j \in \mathcal{I}$. X *est un* **itemset** *(ou un* $k-$**itemset***). Soit* $\mathcal{T} = \{t_1, t_2, ..., t_m\}$ *un ensemble d'estampilles, sur lesquelles un ordre linéaire* $<_T$ *est défini, et où* $t_i <_T t_j$ *signifie que* t_i *précède* t_j. *Une* **transaction** T

est un couple $T = (tid, X)$ *ou tid est l'identifiant de la transaction et* X *est l'itemset associé. À chaque item* i *de* X *est associé l'estampille* t_i *qui représente la date d'apparition de* i *dans* T.
Une transaction $T = (tid, I)$ *supporte un itemset* $X \in \mathcal{I}$ *si* $X \subseteq I$. *Une* **base de transactions** D *est un ensemble de transactions. La* **couverture** *d'un itemset* X *sur* D *est l'ensemble des identifiants de transactions dans* D *qui supportent* X : $couverture(X, D) = \{tid/(tid, I) \in D, X \in I\}$. *Le* **support** *d'un itemset* X *dans* D *est le nombre de transactions dans la couverture de* X *sur* D : $support(X, D) = |couverture(X, D)|$. *La* **fréquence** *d'un itemset* X *sur* D *est le rapport entre la taille de la couverture de* X *sur* D *et la taille de* D : $fréquence(X, D) = \frac{support(X,D)}{|D|}$. *Soit* $\gamma \in]0..1]$ *le support minimum donné par l'utilisateur, un itemset* X *est dit* **fréquent** *si* $fréquence(X, D) \geq \gamma$.

Définition 6 *L'ensemble* F *des itemsets fréquents de* D *avec un support minimum* γ *est noté* $F(D, \gamma) = \{X \in \mathcal{I}/frequence(X, D) \geq \gamma\}$.

Étant donné un ensemble d'items $\mathcal{I}$, une base de transactions D et un support minimum γ, le problème de **l'extraction d'itemsets fréquents** vise à trouver $F(D, \gamma)$ ainsi que le support des itemsets de F. L'exemple 2 donne une illustration des concepts définis dans cette section.

Exemple 2 *La figure 3.1 montre un exemple de base de données* D. *Pour simplifier la lecture, nous supposons que les transactions de* D *sont affichées par ordre de date (i.e.* T_1 *est enregistrée avant* T_2, *etc.) et qu'une estampille unique est associée à tous les items d'une transaction (alors que dans la définition 5 chaque item est estampillé). Avec* $\gamma = \frac{1}{2}$, *les items fréquents (en gras dans les transactions de la figure 3.1) sont* a, b *et* c. *Les itemsets fréquents de* D *sont* (a), (b), (c), *avec un support de* $\frac{6}{10}$, *et* (a, c), *avec un support de* $\frac{1}{2}$.

Notre problème est basé sur les estampilles et vise à extraire des itemsets qui sont fréquents sur des périodes particulières de D. Nous présentons maintenant les notions d'itemset temporel et d'itemset compact, qui sont au cœur de la section 3.5.

Définition 7 *Une* **période** $P = (P_s, P_e)$ *est définie par une date de départ* P_s *et une date de fin* P_e. *L'ensemble des transactions qui appartiennent à une période* P *est défini par* $Tr(P) = \{T/T \subseteq D, \forall i \in T, P_s \leq P_i \leq P_e\}$ *avec* P_i *l'estampille de l'item* i *dans la transaction* T. *Enfin,* PR *est l'ensemble des périodes possibles sur* D.

Tid	items	F(D,1/2)
1	**a b c**	
2	**a c** d	
3	**b** e f	
4	**c** j h	(a)
5	**a** i j	(b)
6	**b** k l	(c)
7	**a b c**	(a c)
8	m n o	
9	**a b c**	
10	**a b c**	

FIG. 3.1 – itemsets fréquents sur D avec $\gamma = \frac{1}{2}$

En d'autres termes, l'ensemble des transactions qui appartiennent à P est l'ensemble des transactions dont tous les items sont estampillés dans les limites de P.

Définition 8 *Un* **itemset temporel** x *est un tuple* (x_i, x_p, x_σ) *où* x_i *est un itemset,* x_p *est une période associée à* x_i *et* x_σ *est le support de* x_i *sur* x_p*. Soit* k *la taille de* x_i*, alors* x *est un* k*-itemset temporel.*

Soit γ, le support minimum, nous proposons la notion d'itemset compact avec la définition 9.

Définition 9 *Soit* x *un itemset temporel.* x *est un* **itemset compact (IC)** *ssi les conditions suivantes sont respectées :*
1) $x_\sigma \geq \gamma$
2) $\forall p_2 \in PR / x_p \subseteq p_2$ *alors on observe a) ou b) ou les deux :*
a) $support(x_i, p_2) < \gamma$
b) $couverture(x_i, p_2) = couverture(x_i, x_p)$
3) $\forall p_2 \in PR / p_2 \subseteq x_p$, $couverture(x_i, p_2) < couverture(x_i, x_p)$
Soit k *la taille de* x_i*, alors* x *est un* **k-itemset compact**. *Enfin,* $\mathbf{SI_k}$ *est l'ensemble de tous les* k*-itemsets compacts.*

La première condition de la définition 9 assure que x représente un itemset qui est fréquent sur sa période. La seconde condition assure que la taille de x_p est maximale. En fait, si une période plus grande existe, alors, sur cette période, x_i n'est pas fréquent ou la couverture de x_i reste identique (*i.e.* étendre la période de x_p à p_2 n'apporte rien au support). Enfin, la troisième condition assure que la taille de x_p est également minimale. En effet,

si x_i est supporté par la première et la dernière transaction de x_p, alors si il existe un période plus petite sur laquelle x_i est fréquent, la couverture sera plus faible (*i.e.* passer de x_p à p_2 implique d'ignorer des transactions qui supportent x_i et doivent donc être gardées). Une illustration de cette définition est donnée dans l'exemple 3.

Exemple 3 *La figure 3.2 montre les k-itemsets compacts qui sont extraits avec* $\gamma = 0,5$. *On peut y constater que les itemsets compacts de taille 1 sur toute la base sont* (a), (b) *et* (c), *que leur support est de* $\frac{6}{10}$, *et que leur période correspond à la base D entière. On peut également observer l'apparition de l'itemset compact* (j)*de taille 1, avec une période restreinte aux dates 3 et 4 et une support de 100%. Ensuite, on peut observer trois itemsets compacts de taille 2 :*

- $(a\ c)$, *avec un support de* $\frac{5}{10}$ *et une période qui couvre toute la base D.*
- $(a\ b)$ *et* $(b\ c)$, *sur la période* [7..10] *avec un support de* $\frac{3}{4}$.

Finalement, il y a un itemset compact de taille 3 : $(a\ b\ c)$ *qui apparaît dans la période* [7..10] *avec un support de* $\frac{3}{4}$. *On peut observer que, grâce à la définition des itemsets compacts, un nouveau type de connaissance émerge. Cette connaissance peut concerner des comportements ponctuels d'utilisateurs. Par exemple, sur D, il s'agit d'un itemset compact de taille 3 (i.e.* $(a\ b\ c)$*) sur une période très précise (i.e.* [7..10]*). Cet itemset, associé à sa période, est optimal au sens de la définition 9 dans la mesure où :*

- *L'itemset est fréquent sur cette période.*
- *Toute période plus grande implique soit la perte du support, soit une couverture identique.*
- *Toute période plus courte, sur laquelle cet itemset est fréquent, ne serait pas maximale.*

Considérons, en effet, l'itemset temporel $y = ((a\ b\ c), [9..10], 100\%)$ *alors* y_i *(l'itemset de y) respecte le support minimum sur sa période, mais il existe une période* $p_2 = [7..10]$ *sur laquelle* $(a\ b\ c)$ *est fréquent avec une couverture plus grande. La condition 2 (maximalité) de la définition 9 n'est pas respectée et y n'est pas un itemset compact. Considérons maintenant* $z = ((a\ b\ c), [6..10], \frac{3}{5})$. *On constate que* z_i *est fréquent sur sa période mais il est également fréquent sur* p_2 *avec une couverture identique, donc* z_i *ne respecte pas la condition 3 de la définition 9 (minimalité) et n'est pas un itemset compact.*

Enfin, notons que les itemsets $(a\ b)$, $(b\ c)$ *et* $(a\ b\ c)$ *n'étaient pas fréquents sur la base D de l'exemple 2 avec* $\gamma = \frac{1}{2}$, *étant donné que leur support sur D est de* $\frac{4}{10}$. *Cependant, grâce à la définition des itemsets compacts, ils peuvent*

être découverts et associés à leur période de fréquence optimale et à leur support sur cette période.

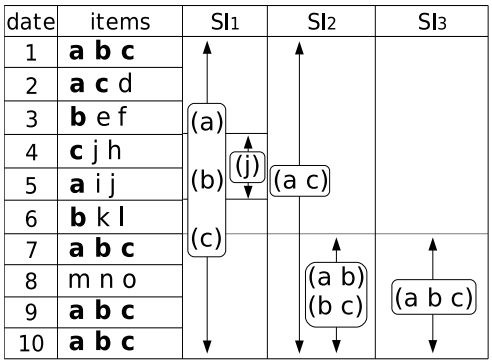

FIG. 3.2 – Itemsets compacts de D avec $\gamma = \frac{1}{2}$

Définition 10 *L'ensemble des* **Itemsets Compacts Maximaux (ICM)** *est défini comme suit : soit x un IC, x est un ICM si les conditions suivantes sont respectées :*
$\forall y \in SI/x \neq y$ *si* $x_i \subseteq y_i$ *alors* $x_p \neq y_p$.

Dans la section 3.5, nous proposons un algorithme optimisé pour la découverte de l'ensemble des ICM, comme décrits par la définition 10.

Trouver les motifs de l'exemple 3 ne se limite pas à une simple baisse du support. D'abord, parce qu'un support trop faible est une source bien connue d'échec pour les algorithmes de fouille. Ensuite, parce que notre objectif est aussi d'associer les itemsets compacts à leurs périodes de fréquence. Il s'agit d'un nouveau type de connaissance qui informe sur la fenêtre temporelle dans laquelle un itemset est fréquent. Il ne suffit pas non plus de diviser la base en plusieurs sous-ensembles afin de découvrir des comportements sur certaines périodes. Cela aurait pour conséquence de casser des périodes possibles (elles sont à découvrir, donc on ne peut pas savoir où faire le découpage) et donc d'ignorer les comportements qui leur sont associés. Enfin, une motivation importante de ce travail vient du fait que le nombre total de combinaisons possibles pour les itemsets compacts est de $(2^n \times k!)$, avec n le nombre d'itemsets et $k = |D|$. Donc, 2^n est le nombre d'itemsets potentiels sur D et $k!$ le nombre de fenêtre temporelle possibles sur D. La propriété de monotonicité nous évite l'énumération de tous les itemsets possibles. Nous l'utiliserons également pour éviter l'énumération de toutes les fenêtres temporelles.

3.3 Travaux existants

Dans la littérature, nous trouvons de nombreux travaux sur les aspects temporels liés aux règles d'association. Dans cette section, je propose d'étudier des méthodes qui ont pour but la découverte d'itemsets temporels dans des données statiques. Plus précisement, cette temporalité peut s'exprimer dans quatre catégories différentes :

1. **Une période précise est données** et le but est de trouver les itemsets fréquents dans cette période. Dans [AR00] les auteurs proposent la notion de règle d'association temporelle. Leur idée consiste à extraire les itemsets qui sont fréquents sur une période précise qui sera plus courte que la base de données complète. Les périodes proposée par [AR00] sont définies par la durée de vie de chaque item. Ainsi, le processus de data mining qui va extraire les motifs se contraint aux périodes définies par ces durées de vie. Une idée similaire est proposée [LLC01] où les auteurs proposent d'extraire des itemsets dans une base de publications. Leur but est de découvrir des règles de la forme $(X \Rightarrow Y)^{t,n}$, où t correspond à la première apparition de X et de Y dans la même transaction et n correspond à la fin de la base de données. Notons également les travaux de [SLRdATdC09] qui propose une classification des usages du Web attentive à l'évolution. Les auteurs détaillent en premier lieu les effets négatifs d'une exploration sur l'ensemble des données (dont le résultat ne reflète que les comportements les plus importants sur toutes les données). Ensuite, les auteurs proposent de considérer les données par périodes successives (un mois, par exemple) afin d'en extraire des classes de comportements plus précises et d'observer leur évolution.

2. **Un motif spécifique est donné** et le but est de trouver la période qui correspond. Dans [CP99] les auteurs proposent d'identifier les périodes valides et la périodicité des motifs. En d'autres termes, étant donnée une règle d'association spécifique donnée par l'utilisateur, le but est de trouver un intervalle qui valide cette règle. Dans [YZS05] les auteurs proposent d'extraire des associations à "profil-temporel" dans le but de découvrir des relations intéractives cohérentes par rapport à une requête.

3. **Extration de motifs périodiques (récurrents).** Dans cette catégorie, les estampilles sont analysées dans le but de trouver des motifs

répétitifs, *i.e.* des motifs qui apparaissent de manière régulière et avec une périodicité précise dans les enregistrements. Dans [LNWJ03], un modèle de répétition est donné, inspiré par le calendrier. Par exemple, $< 2000, *, 16 >$ correspond au 16^{eme} jour du mois en 2000. Ils proposent alors d'extraire les règles d'association avec ces modèles de répétition pour contrainte selon deux méthodes : une précise et l'autre floue. Dans [ORS98], on trouve une définition du problème de la fouille de règles d'associations cycliques. Une règle est considérée comme cyclique si elle respecte le support et la confiance donnés par l'utilisateur à des intervalles de temps réguliers. Deux algorithmes sont proposés par [ORS98] pour résoudre ce problème d'extraction.

L'objectif de nos travaux est de proposer une nouvelle catégorie : la découverte de motifs associés à leurs périodes de fréquence optimale.

Des outils [Web, hA] existent à l'heure actuelle pour analyser les logs avec des niveaux de granularité variables (jour, mois, année). Ils permettent par exemple de connaître l'évolution du nombre de clients sur le site ou du nombre de requête pour chaque page. Cependant, ces outils dépendent du découpage qui est fait des données, de la granularité choisie et ils ne sont pas en mesure de proposer des "schémas" au sens de la définition de l'ECD (comme les motifs séquentiels par exemple). Notre proposition va au delà des notions de comptage afin de proposer une extraction de connaissance sur les logs d'accès Web. Plus proche de notre problématique, [MR04] propose une méthode d'extraction de règles d'épisodes dans une longue séquence d'événements est présentée. Les auteurs proposent également de déterminer la taille de fenêtre optimale pour le support des motifs extraits. Ces travaux se situent cependant dans le contexte d'une longue séquence de données. Notre proposition s'inscrit dans la recherche de motifs séquentiels à partir d'une base de séquences et permet d'extraire les périodes qui contiennent des motifs fréquents. Les algorithmes proposés seront par nature différents car dans le cas de [MR04], le support correspond au nombre d'occurrences du motif dans la séquence analysée (par opposition au support de la définition 3, qui se base sur le nombre de séquences de la base qui contiennent le motif).

3.4 Extraction de périodes contenant des motifs séquentiels fréquents

Cette section est basée sur l'article « Web usage mining : extracting unexpected periods from web logs », Florent Masseglia, Pascal Poncelet, Maguelonne Teisseire et Alice Marascu, publié dans la revue internationale Data Mining and Knowledge Discovery (DMKD) en 2008.

Cette section présente un premier travail sur le thème de l'évolution en tant que connaissance. Il s'agit d'une heuristique permettant d'extraire les motifs fréquents sur des périodes calculées en fonction des connexions sur le site. Quand une période ne connait aucune connexion ni déconnexion (les utilisateurs sont les mêmes pendant cette période) alors elle est considérée comme stable. L'idée qui motive cette heuristique est que les utilisateurs connectés en même temps ont plus de chances de partager des objectifs de navigation communs. La motivation principale étant surtout de valider l'existence de périodes dans lesquelles des motifs deviennent fréquents alors qu'ils ne le sont pas avec une analyse des données globale.

3.4.1 Motivations

Dans les grandes lignes, notre objectif est d'énumérer l'ensemble des périodes issues du log à analyser afin de déterminer quelles sont celles qui contiennent des motifs séquentiels fréquents. Nous définirons dans cette section les notions de période et de motifs séquentiel fréquent sur une période. Considérons l'ensemble de transactions issu de la figure 3.3 (tableau de gauche). Ces transactions sont ordonnées par date croissante, ce qui est le cas dans les logs. On peut y observer que le client c_1 s'est connecté à la date 1 pour demander l'URL a et que le log contient 9 enregistrements (jusqu'au client c_3 qui s'est connecté à la date 9 pour demander l'URL f). Considérons maintenant les dates "d'entrée" et de "sortie" de chaque client (figure 3.3, premier tableau de droite). La première action du client c_1 s'est produite à la date 1, et sa dernière action enregistrée dans le log apparaît à la date 4. Ainsi on peut en déduire les différentes périodes de ce log en matière de clients connectés. On distingue 5 périodes. Pendant la première période (de la date 1 à la date 2) le client c_1 était le seul connecté sur le site. Puis les clients c_1 et c_2 sont connectés ensembles sur la période p_2 (de la date 3 à la date 4), etc.

Considérons à présent les séquences des clients du log de la figure 3.3. Ces séquences sont représentées dans la figure 3.4, ainsi que les motifs trouvés

Client	date	URL
c_1	1	a
c_1	2	b
c_2	3	a
c_1	4	d
c_2	5	d
c_3	6	d
c_2	7	e
c_3	8	e
c_3	9	f

Client	entrée	sortie
c_1	1	4
c_2	3	7
c_3	6	9

Période	Intervalle	Clients
p_1	[1..2]	c_1
p_2	[3..4]	$c_1,\ c_2$
p_3	[5]	c_2
p_4	[6..7]	c_2, c_3
p_5	[8..9]	c_3

FIG. 3.3 – Un log de trois clients et les périodes associées

sur la totalité du log et sur les différentes périodes. Avec un support de 100% et sur la totalité du log, le seul motif séquentiel que l'on trouve pour ces 3 séquences est le motif : $< (d) >$, qui ne contient que l'item d. A présent, considérons les différentes périodes identifiées plus haut, ainsi que les clients connectés à chaque période. Pour les périodes p_1, p_3 et p_5, qui ne concernent qu'un seul client, on ne trouve qu'un seul motif, correspondant à la séquence du client. Pour la période p_2 on trouve le motif $< (a)\ (d) >$ (motif séquentiel commun aux deux seuls clients connectés pendant la période p_2 : c_1 et c_2). Enfin, pour la période p_4, on trouve le motif $< (d)\ (e) >$.

Client	Séquence	log	$p_1,\ p_3,\ p_5$	p_2	p_4
c_1	$< (a)\ (b)\ (d) >$	$< (d) >$	–	$< (a)\ (d) >$	$< (d)\ (e) >$
c_2	$< (a)\ (d)\ (e) >$		–		
c_3	$< (d)\ (e)\ (f) >$		–		

FIG. 3.4 – Les motifs séquentiels fréquents pour les clients connectés à chaque période

Plus formellement, nous définissons dans la suite de cette section, les notions de période, de clients connectés et de mouvement. Soit C l'ensemble des clients du log et D l'ensemble des dates enregistrées.

Définition 11 *L'ensemble P des périodes possibles sur le log est défini de la manière suivante :*

$P = \{(p_a, p_b)/(p_a, p_b) \in D \times D \; et \; a \leq b\}$.

Dans la définition suivante, nous considérons que $d_{min}(c)$ et $d_{max}(c)$ sont les dates d'entrée et de sortie de c dans le log (première et dernière action enregistrées pour c).

Définition 12 *Soit $C_{(a,b)}$ l'ensemble des clients connectés pendant la période (a, b). $C_{(a,b)}$ est défini de la manière suivante :*
$C_{(a,b)} = \{c/c \in C$
$et \; [d_{min}(c)..d_{max}(c)] \cap [a..b] \neq \emptyset\}$.

Enfin, nous définissons les notions de *période de mouvement* et de *période dense*. Dans le premier cas, il s'agit de chaque période maximale p_m sur laquelle C_{p_m} ne varie pas. Avec l'exemple donné dans la figure 3.3, la période [6..7] est une période de mouvement. Ce n'est, en revanche, pas le cas de [3..3] car elle est incluse dans [3..4] qui contient exactement les mêmes clients (*i.e.* $C_{(3,3)} = C_{(3,4)}$). Une *période dense* est une période de mouvement qui contient au moins un motif séquentiel fréquent. Dans l'exemple donné en introduction, la période correspondant au 31 janvier, pendant la séance de TP, devrait être une période dense.

Définition 13 *Soit P_{mouv} l'ensemble des périodes de mouvements, P_{mouv} est défini de la manière suivante :*
$P_{mouv} = \{(m_a, m_b)/(m_a, m_b) \in P \; et$

1) $\nexists \; (m'_a, m'_b)/(b - a) < (b' - a')$
$et \; [a'..b'] \cap [a..b] \neq \emptyset$
$et \; C_{(m'_a, m'_b)} = C_{(m_a, m_b)}$

2) $\forall (x, y) \in [a..b], \forall (z, t) \in [a..b]/$
$x \leq y, z \leq t \; on \; a \; C_{(x,y)} = C(z, t).\}$

Dans la définition 13, la condition 1 exprime le fait qu'il n'existe pas de période plus grande qui soit en "contact" avec P_{mouv} et qui concerne les mêmes clients. La condition 2 exprime le fait que toutes les périodes de P_{mouv} concernent les mêmes clients.

Définition 14 *Une période de mouvement p est dite dense si C_p contient au moins un motif séquentiel fréquent respectant le support minimum spécifié par l'utilisateur proportionnellement à $|C_p|$.*

La notion de période dense (définition 14), est au cœur des travaux présentés dans cette section. Dans la suite, notre objectif est d'extraire ces périodes à partir du fichier log. Pour illustrer cette définition, considérons une période p_e concernant 100 clients ($|C_{p_e}| = 100$) et un support minimum de 5%, tout motif séquentiel inclus dans au moins 5 navigations de C_{p_e} sera considéré comme fréquent pour cette période. Si il existe au moins un motif fréquent dans p_e, alors p_e devra être extraite par notre méthode. Extraire les motifs séquentiels fréquents sur chacune de ces périodes avec une méthode classique n'est pas une solution envisageable pour les raisons suivantes :

- Les algorithmes d'extraction de motifs séquentiels (tels que PSP [MCP98b] par exemple) peuvent être mis en défaut si le motif à extraire est très long. Dans le cas des accès Web, il n'est pas rare de trouver plusieurs dizaines d'occurences d'une même requête pour un client (*e.g.* les fichiers pdf ou php). Dans le cas où le support de ce phénomène dépasserait le support minimum sur C_p, aucune méthode d'extraction de motifs séquentiels exhaustive existante ne pourrait obtenir les résultats.
- Lors de nos expérimentations, avec 14 mois de log, nous avons obtenu un total de $3,5$ millions de périodes de mouvement environ. Nous pensons qu'il est plus pertinent d'obtenir des périodes denses approchant au mieux le résultat réel en passant par une heuristique, plutôt que faire appel plusieurs millions de fois à un algorithme de fouille de données exhaustif.

La section suivante présente l'idée générale de notre approche afin de fournir les périodes de mouvement denses qui sont détectées à partir des données enregistrées dans le log.

3.4.2 Principe général

La figure 3.5 donne une vue d'ensemble de l'heuristique PERIO que nous avons développée pour résoudre le problème de l'extraction de périodes denses et des motifs séquentiels associés. Tout d'abord, à partir du log, les périodes sont construites par la phase de prétraitement décrite en section 3.4.3. Ces périodes sont ensuites considérées une par une, par ordre croissant de date de début. Pour chaque itération n, la période p_n est considérée. L'ensemble des clients C_{p_n} est chargé en mémoire à partir du log ("DB" dans la figure 3.5). Les items fréquents de C_{p_n} sont alors utilisés (opération notée "1"

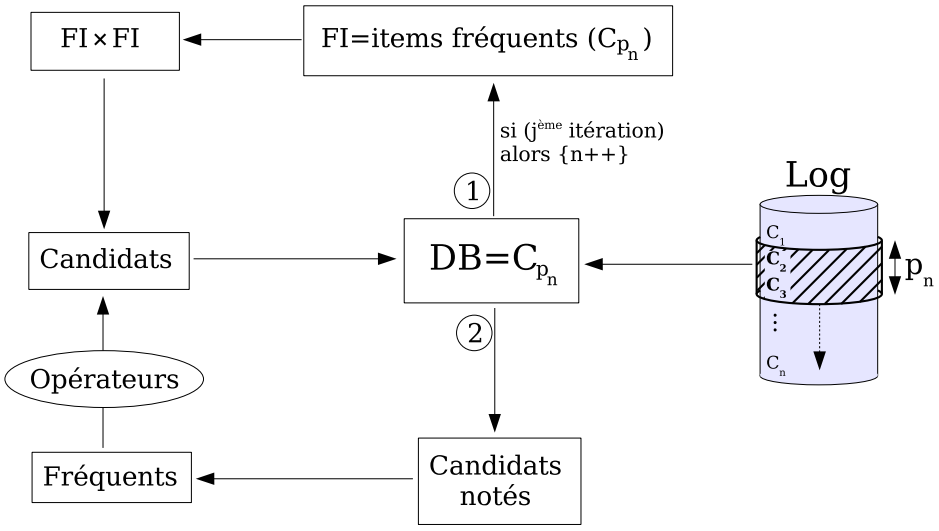

FIG. 3.5 – Vue d'ensemble des opérations effectuées par PERIO

dans la figure 3.5) pour produire des candidats de taille 2 par auto-jointure[1]. Les candidats sont ensuite comparés avec les séquences de C_{p_n} afin de déterminer les fréquents (étape notée "2" dans la figure 3.5). Les fréquents sont utilisés par les opérateurs de voisinage décrit en section 3.4.3 et les nouveaux candidats sont comparés avec les séquences de C_{p_n}. Afin d'obtenir des résultats plus fins sur chaque période, il est possible de spécifier un nombre d'itération minimum (j) au delà duquel n est incrémenté pour passer à la période suivante.

3.4.3 Une heuristique pour l'extraction des périodes denses

Nous décrivons dans cette section les étapes permettant d'obtenir les périodes denses à partir d'un log d'accès Web. Ces étapes vont du prétraitement à la visualisation des résultats. Nous décrivons aussi les opérateurs de voisinage mis en place pour l'heuristique PERIO.

Nous n'abordons pas, dans cette section, les problèmes liés au prétraitement d'un fichier log dans le sens généraliste. Nous considérons qu'une

[1] Notons que cette opération n'est effectuée qu'une fois sur n, avec n défini par l'utilisateur, car elle génère un très grand nombre de candidats.

technique de prétraitement (comme [TT04]) visant à identifier les navigations, les sessions, les robots et clients "parasites" a déjà été appliquée sur ce log. Le prétraitement que nous mettons en place est trivial. Il s'agit de suivre les étapes de la figure 3.3. Pour chaque client, nous identifions sa date d'entrée et sa date de sortie dans le log. Ensuite, les dates sont enregistrées au format "date, action, client" qui exprime pour chaque date si il s'agit d'une entrée ou d'une sortie ("action") et quel est le client concerné. Ces dates sont ensuite triées par ordre croissant et enregistrées dans un historique. La lecture de cet historique permet de savoir, pour chaque date lue, l'action à effectuer (ajout ou suppression de la séquence du client concerné dans "DB", en mémoire).

Dans la mesure où notre proposition est basée sur une heuristique, notre but est de fournir un résultat répondant aux critères suivants :

Pour chaque période p appartenant à l'historique du log, soit $resultatReel$ le résultat à obtenir (le résultat qu'obtiendrait un algorithme de fouille de données qui explore tout l'ensemble des solutions après analyse des clients de C_p). $resultatReel$ est alors l'ensemble des motifs séquentiels à trouver. Soit $resultatPer$ les résultats obtenus par la méthode proposée dans cette section. Nous voulons minimiser $\sum_{i=0}^{taille(resultatPer)} S_i/S_i \notin resultatReel$ tout en maximisant $\sum_{i=0}^{taille(resultatReel)} R_i/R_i \subseteq resultatPer$.
En d'autres termes, nous voulons trouver toutes les séquences apparaissant dans $resultatReel$ tout en évitant que le résultat soit plus grand qu'il ne le devrait (sinon l'ensemble de toutes les séquences, de toutes les navigations, pourrait constituer un résultat, car il englobe le résultat réel).

Cette heuristique emprunte aux algorithmes génétiques leur conception du voisinage, en y intégrant les propriétés des motifs fréquents pour optimiser les candidats proposés.

La principale idée, sur laquelle PERIO se base, consiste à parcourir l'ensemble P_{mouv} des périodes de mouvement et, pour chaque période p de P_{mouv}, à générer des populations de candidats grâce aux items fréquents et aux opérateurs de voisinage. Ensuite ces candidats sont comparés avec les séquences de C_p afin d'évaluer leur pertinence (ou tout au moins leur distance d'une séquence fréquente). Ces deux phases (opérateurs de voisinage et évaluation des candidats) sont expliquées dans cette section.

Opérateurs de voisinage

Les opérateurs de voisinage présentés dans cette section, ont été validés grâce à une batterie d'expérimentations réalisées sur les logs de l'Inria So-

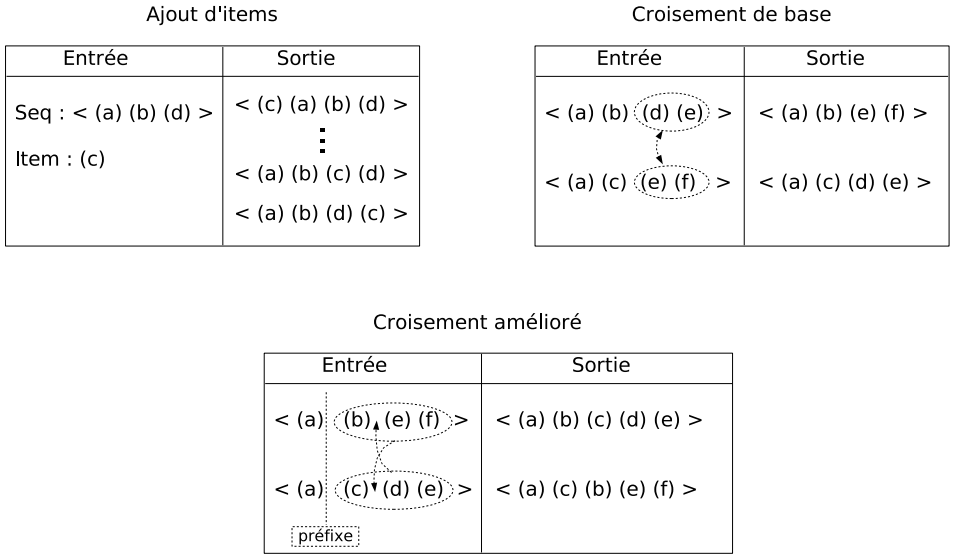

FIG. 3.6 – Quelques opérateurs conçus pour la recherche de séquences de navigation fréquentes

phia Antipolis. Nous avons choisi des opérateurs de type génétique, aussi bien que des opérateurs basés sur les propriétés des motifs séquentiels fréquents. Quand nous faisons référence aux "séquences choisies aléatoirement" (ou bien "séquences aléatoires"), nous utilisons une roulette biaisée, telle que les séquences ayant le support le plus haut sont choisies plus fréquemment que les séquences de support plus faible.

Enfin, nous nous sommes efforcés d'évaluer le taux de réussite de nos opérateurs, en calculant la moyenne des séquences retenues par rapport aux séquences proposées, en fin de processus. Ainsi, un opérateur affichant un taux de réussite de 20% est un opérateur qui, en fin de processus, a vu 20% des candidats qu'il propose s'avérer fréquents. Le taux de réussite brut, est un taux de réussite duquel on a décompté les séquences proposées par d'autres opérateurs et déterminées fréquentes (*i.e.* les séquences fréquentes trouvées uniquement grâce à cet opérateur).

Nouveaux items fréquents :
Quand un nouvel item fréquent apparaît sur C_p il est utilisé pour générer

toutes les séquences de taille 2 possibles avec les autres items fréquents. Les candidats ainsi générés sont ajoutés à l'ensemble des candidats à tester. En raison du grand nombre de candidats générés, cet opérateur n'a qu'un taux de succés de 15%. Cet opérateur reste cependant essentiel car il permet d'obtenir toutes les séquences fréquentes de taille 2, qui sont la base pour les opérateurs suivants. Il n'est cependant pas utilisé à chaque mouvement, en raison du trop grand nombre de candidats générés. Nous avons donc limité les appels à cet opérateur pour qu'il soit invoqué chaque n mouvement dans la lecture de l'historique.

Ajout d'items :
Cet opérateur a pour but de choisir un item aléatoirement parmi les items fréquents, puis d'ajouter cet item à une séquence s choisie aléatoirement, après chaque item de s. Chaque ajout donnant lieu à un nouveau candidat. Cet opérateur génère $longueur(s) + 1$ candidats. Par exemple, avec la séquence < (a) (b) (d) > et l'item c, nous allons générer les séquences candidates < (c) (a) (b) (d) >, < (a) (c) (b) (d) >, < (a) (b) (c) (d) > et finalement < (a) (b) (d) (c) >. Cet opérateur connait un taux de réussite de 20%, mais les séquences trouvées sont très utiles au reste des opérateurs.

Croisement de base :
Cet opérateur (largement inspiré des opérateurs génétiques) utilise deux séquences aléatoires afin de proposer deux candidats issus de leur croisement. Par exemple, avec les séquences < (a) (b) (d) (e) > et < (a) (c) (e) (f) >, nous proposons les candidats < (a) (b) (e) (f) > et < (a) (c) (d) (e) >. Cet opérateur affiche un taux de réussite performant (50%) grâce aux séquences fréquentes contenues dans les séquences obtenues par les opérateurs précédents.

Croisement amélioré :
Ce nouvel opérateur, conçu pour être une amélioration du croisement de base, repose sur les propriétés des séquences fréquentes. Cet opérateur a pour but de choisir deux séquences aléatoires, et le croisement ne s'effectue pas au milieu des séquences, mais à la fin du plus long préfixe commun aux deux séquences traitées. Considérons deux séquences < (a) (b) (e) (f) > et < (a) (c) (d) (e) > issues du croisement précédent. Le plus long préfixe commun à ces deux séquences est < (a) >. Le croisement va donc commencer après l'item qui suit a, pour chaque séquence. Dans notre exemple, les deux séquences résultant de ce croisement seront < (a) (b) (c) (d) (e) > et < (a) (c) (b) (e) (f) >. Cet opérateur connait un taux de succès de 35%.

Précisons que les séquences déjà obtenues grâce au croisement de base sont décomptées lors du calcul du taux de réussite de cet opérateur. Ce taux de 35% est donc un apport brut pour les opérateurs précédent (*i.e.* des séquences que cet opérateur est le seul à fournir).

Dernier croisement :
Un dernier opérateur de croisement consiste à améliorer les deux précédents. Il est basé sur le même principe que l'opérateur de croisement amélioré, à cette différence que la seconde séquence n'est pas choisie aléatoirement. En effet la seconde séquence est choisie comme étant celle qui présente le plus long préfixe commun avec la première séquence choisie. Cet opérateur présente un taux de réussite (brut) de 30%.

La figure 3.6 donne une illustration de quelques opérateurs décrits dans cette section.

L'heuristique PERIO est décrite par l'algorithme donné en figure 3.7 :

Evaluation des candidats

Pour chaque période de P_{mouv}, PERIO génère les nouveaux candidats et compare ensuite chacun des candidats aux séquences de C_p. La comparaison consiste à obtenir un pourcentage qui représente la distance entre la séquence s du candidat et chaque séquence c de C_p. Si s est incluse dans c, le pourcentage sera de 100% et ce taux va décroitre avec l'apparition de "parasites" (différences entre le candidat et la séquence de navigation). Pour évaluer cette distance, le pourcentage est obtenu en divisant la taille de la plus longue séquence commune (PLSC [CLR]) entre s et c par la taille de s. Par exemple, si s, de taille 4, contient une sous-séquence de taille 3 en commun avec c, alors la note de s pour c sera de 3/4. De plus, dans le but d'obtenir des séquences les plus longues possible, nous utilisons un algorithme qui favorise les séquences les plus grandes, si elles sont incluses dans c. D'un autre côté, l'algorithme sanctionne les séquences trop longues si elles ne sont pas incluses (plus la séquences est longue, plus sa distance à la séquence de navigation sera pénalisante).

Pour prendre en compte tous ces paramètres, le calcul effectué pour comparer les candidats à chaque séquence de C_p est décrit par l'algorithme NOTER décrit en figure 3.8.

Algorithme *Perio*
In : P_{mouv} l'ensemble des périodes de mouvement.
Out : SP les motifs séquentiels correspondants
aux comportements fréquents.
For $(p \in P_{mouv})$ {
// Maintenir le support des items
itemsSupports=getItemsSupports(C_p) ;
// Générer des candidats à partir des items
// et des motifs fréquents
candidats=voisinage(SP, *pagesSupport*) ;
For $(c \in candidats)$ {
For $(s \in C_p)$ {
Noter(c, s) ;
}
}
For $(c \in candidats)$ {
If (support(c) $>$ *minSupport* `OU` *critere*){
inserer(c, SP) ;
}
}
}
Fin algorithme *Perio*

FIG. 3.7 – L'heuristique PERIO

Enfin, les candidats évalués par l'algorithme NOTER seront insérés dans SP si leur support est supérieur au support minimum ou si ils correspondent à un "critère de sélection naturelle". Ce critère, spécifié par l'utilisateur prend la forme d'un pourcentage qui définit la distance entre le support du candidat et le support minimum. Dans un nombre de cas (choisis de façon aléatoire), les candidats dont le support correspond au critère peuvent être insérés dans SP afin d'éviter à l'heuristique PERIOD de converger vers un optimum local.

3.4.4 Expérimentations

Les programmes d'extraction sont réalisés en C++ sur des machines de type PC équipés de processus pentium 2,1 Ghz et exploités par un système RedHat. Nous avons effectué nos expérimentations sur les logs de l'Inria Sophia Antipolis. Ces logs sont découpés à raison de un log par jour. A la

Algorithme *Noter*
In : c le candidat à évaluer et s la séquence
de navigation du client.
Out : $p[c]$ le pourcentage affecté à c.
// Si c est incluse dans s, c est favorisée
If ($c \subseteq s$) $p[c]$=100+taille(c) ;
// Si c, de longueur 2, n'est pas incluse alors
// la garder ne présente aucun intérêt
If (taille(c)$\leq$ 2) $p[c]$=0 ;
// Sinon, donner une note à c et
// défavoriser les distances accrues
$p[c]=\frac{taille(PLSC(c,s))*100}{taille(c)} - taille(c)$;
Fin algorithme *Noter*

FIG. 3.8 – L'algorithme NOTER

fin du mois, les logs journaliers sont regroupés sous forme d'un log mensuel. Nous avons donc travaillé sur les 14 logs mensuels disponibles, que nous avons considérés comme un seul log global recouvrant 14 mois d'enregistrements (de janvier 2004 à mars 2005).
Ce log de 14 mois représente environ 14 Go de données. Il contient 3,5 millions de séquences (clients), la longueur moyenne de ces séquences est de $2,68$ et la longueur maximale est de 174 requêtes. Le log contient environ 2 millions de périodes et 300000 items. Le temps d'exécution de PERIOD sur ce log est d'environ 6 heures avec un support minimum de 2% (nous avons trouvé 1981 comportements fréquents qui sont ensuite régroupés en 400 clusters environ).
La figure 3.10 montre l'évolution dans le temps du nombre d'utilisateurs ayant respecté 6 de ces comportements. Le premier graphique de la figure 3.10 montre l'évolution des comportements :

- $C1$ =<(semir/restaurant)
  (semir/restaurant/consult.php)
  (semir/restaurant/index.php)
  (semir/restaurant/index.php)>
- $C2$ =<(eg06) (eg06/dureve_040702.pdf) (eg06/fer_040701.pdf) (eg06)>

Afin de rendre le graphique plus lisible et les "pics" plus évidents, ces évolutions sont vues sur une période qui va de début mai jusqu'à fin juillet. Le comportement $C1$ correspond à une navigation typiquement périodique. En effet, la cantine de l'Inria Sophia Antipolis a été en travaux pendant cette période

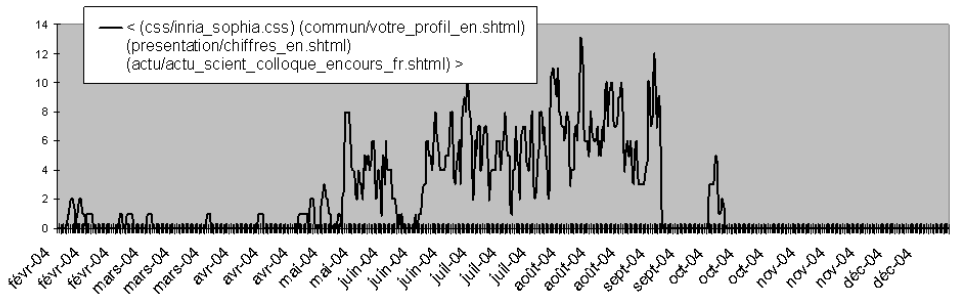

FIG. 3.9 – Un comportement fréquent sur plusieurs semaines consécutives

et les membre de l'unité pouvaient, via les pages web "semir/restaurant", commander leurs repas froids de la semaine. Le comportement $C2$ correspond à une navigation sur des pages relatives aux "états généraux" de la recherche.
Le deuxième graphique de la figure 3.10 montre l'évolution des comportements :

- $C3$ =<(requete.php3) (requete.php3) (requete.php3)>
- $C4$ =<(Hello.java) (HelloClient.java) (HelloServer.java)>

Avec le préfixe "mascotte/anonymisé1/web/td1/" pour $C3$ et "oasis/anonymisé2/ProgRpt/TD03-04/hello/" pour $C4$. Ces deux comportements correspondent à la consultation de pages de TD/TP sur les sites des chercheurs qui les proposent.
Enfin, le troisième graphique de la figure 3.10 montre l'évolution des comportements :

- $C5$ =<(mimosa/fp/Skribe) (mimosa/fp/Skribe/skribehp.css) (mimosa/fp/Skribe/index-5.html)>
- $C6$ =<(sgp2004) (navbar.css) (submission.html)>

Avec le préfixe "geometrica/events/" pour $C6$. Le comportement $C5$ est interprété par l'auteur des pages comme une conséquence de nombreux échanges en mars 2004 sur la mailing-list de Skribe. Le comportement $C6$ connaît deux pics (début avril et mi-avril). Il s'agit d'un comportement relatif à la soumission d'articles pour le symposium sgp2004, dont la date limite de sou-

H

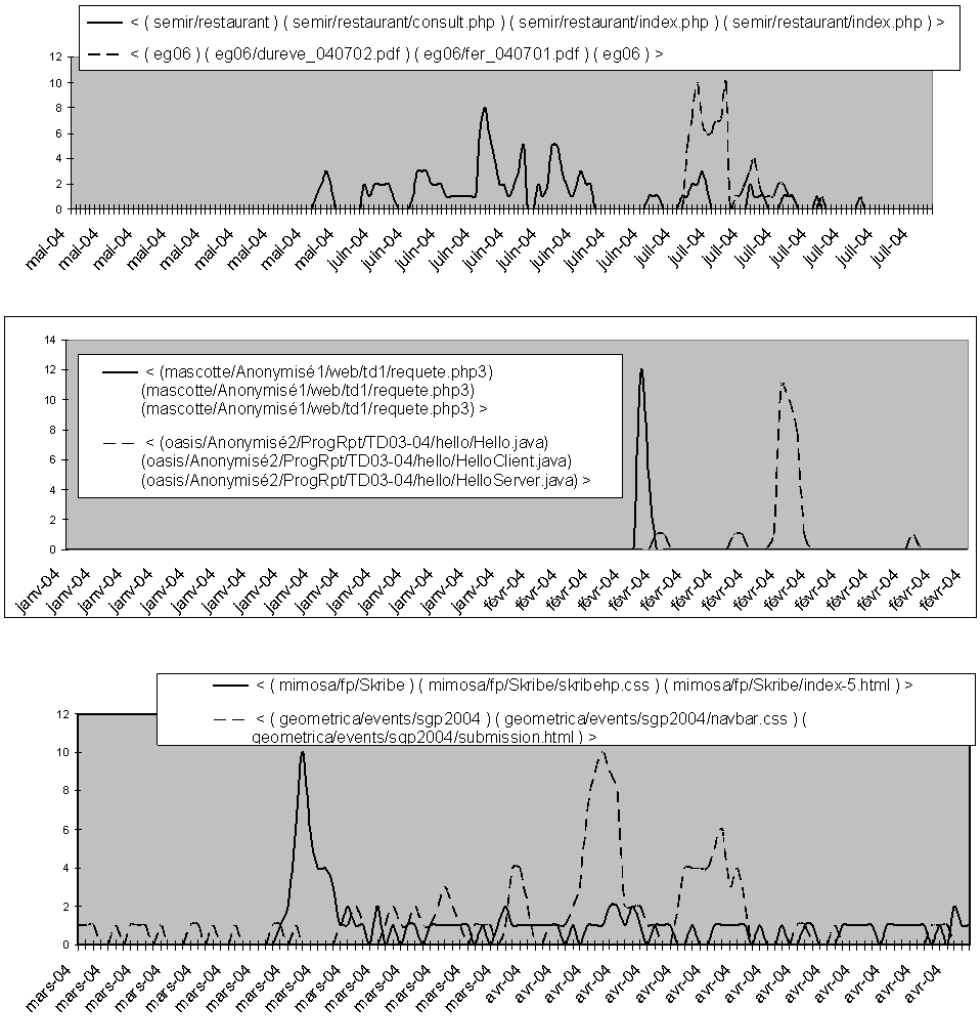

FIG. 3.10 – Pics de fréquences pour 6 comportements

mission était le 7 avril pour les résumés et le 14 avril pour les articles.

Certains des comportements que nous avons mis en évidence n'ont pas un caractère ponctuel et se retrouvent sur plusieurs semaines, voir plusieurs mois. Leur fréquence dans le log est proportionnelle au nombre de clients connectés à chaque période. C'est le cas par exemple de $C7$ =<(css/inria_sophia.css) (commun/votre_profil_en.shtml) (presentation/chiffres_en.shtml) (actu/actu_scient_colloque_ encours_fr.shtml)> dont l'évolution est tracée dans la figure 3.9. On y observe que ce comportement se retrouve principalement sur 5 mois consécutifs (de mai à septembre).

3.5 Extraction de toutes les périodes contenant des itemsets fréquents

Cette section est basée sur l'article « Time Aware Mining of Itemsets » (Bashar Saleh, Florent Masseglia) publié dans les actes de la conférence internationale TIME 2008.

Cette section présente l'algorithme DeICo (Découverte d'Itemsets Compacts). La notion de kernel, présentée dans cette section, permet une extraction précise des itemsets compacts. D'abord, nous donnons une vue générale du principe de cette extraction dans la section 3.5.1. Ensuite, les détails de l'algorithme sont donnés dans la section 3.5.2.

3.5.1 Principe Général

DeICo introduit un nouveau principe de comptage pour les itemsets candidats. Considérons un itemset temporel t qui n'est pas compact (*i.e.* $t_\sigma < \gamma$). Tout surensemble $u = (u_x, u_p, u_\sigma)/t_x \subseteq u_x \wedge u_p \subseteq t_p$ de t ne peut pas être un itemset compact (*i.e.* $u_\sigma < \gamma$). DeICo étend le principe d'apriori afin de générer des itemsets compacts candidats et compter leur support. Le principe de génération est modifié par l'ajout d'un filtre sur les intersections possibles entre candidats (*i.e.* si deux itemsets compacts de taille k ont un préfixe commun mais ne partagent pas la même période, alors leur croisement ne peut pas générer un itemset compact). Cependant, l'étape de comptage d'apriori ne peut pas s'appliquer directement dans notre cas. Considérons c, un itemset temporel candidat.

Une solution consisterait à compter le nombre d'apparitions de c dans c_p. Ce n'est pas une solution correcte. Considérons en effet le candidat $c = ((a\ b), [1..10], c_\sigma)$ (généré à partir de $x = ((a), [1..10], \frac{6}{10})$ et $y = ((b), [1..10], \frac{6}{10})$). c n'est pas compact car $c_\sigma = \frac{4}{10}$. Toutefois, sur c_p, il existe un itemset compact $c' = ((a\ b), [7..10], \frac{3}{4})$. Notre but, pendant le comptage, est de construire des kernels qui correspondent aux périodes de fréquence des itemsets temporels candidats. Ensuite, ces kernels seront fusionnés dans le but d'obtenir les itemsets compacts. La définition 15 précise ces concepts illustrés par la figure 3.13. Cette définition récursive s'adapte bien au fait que l'on effectue des passes successives sur les données afin de trouver les périodes qui correspondent aux itemsets compacts. En effet, la façon dont une passe est effectuée (soit dans l'ordre séquentiel des transactions) implique de découvrir les kernels "à la volée".

Définition 15 *Un* **kernel** *est une période. L'ensemble $K(x, P, \gamma)$ des kernels de l'item x sur la période P pour un support γ est défini comme suit : Soit $k \subseteq P$ une période telle que $x \subseteq Tr(k_s) \wedge Tr(k_s)$ est la première apparition de x sur P. Si k n'existe pas, alors $K = \emptyset$. Sinon, soit N l'ensemble des estampilles telles que $\forall n \in N, n \in P \wedge n > k_s \wedge frequence(x, [k_s..n]) < \gamma$ (en d'autres termes, N est l'ensemble des estampilles de P telles que toute extension de k à une estampille de N implique la perte de fréquence pour x). Si N est vide, alors k_e est défini comme la dernière apparition de x dans P et $K(x, P, \gamma) = \{k\}$. Sinon (i.e. $N \neq \emptyset$), soit $m \in N / \forall n \in N, n > m$ (m est la première estampille telle que la fréquence de x est perdue sur $[k_s..m]$). Alors k_e est défini comme la dernière apparition de x sur $[k_s..m]$ et $K(x, P, \gamma) = \{k\} \cup K(x, P - [k_s..k_e], \gamma)$.*

Exemple 4 *Considérons l'itemset temporel candidat de taille 1, $c = ((b), [1..10], c_\sigma)$. La figure 3.11 donne la table booléenne des apparitions de b. Il y a deux kernels de (b) sur c_p ($[1..3]$ et $[6..10]$). Ces kernels peuvent être fusionnés pour obtenir un itemset compact $((b), [1..10], \frac{6}{10})$.*

date	b	kernels	merge
1	1	Kernel 1: [1..3] threshold=2/3	Itemset: (b) period: [1..10] threshold: 6/10
2	0		
3	1		
4	0		
5	0		
6	1	Kernel 2: [6..10] threshold=4/5	
7	1		
8	0		
9	1		
10	1		

FIG. 3.11 – Kernels et période de l'itemset (b)

Considérons l'itemset x et K l'ensemble des kernels de x sur la période P avec un support minimum γ. Grâce au lemme 1 nous montrons que l'algorithme MERGEKERNELS (figure 3.5.1) permet d'obtenir les itemsets compacts de x sur P qui respectent γ.

Lemme 1 *Soit K l'ensemble des kernels de x sur P avec un support minimum σ. L'algorithme* MERGEKERNELS *permet d'extraire les itemsets compacts $s = (x, x_p, \sigma)$ sur P avec le support γ.*

Algorithm MergeKernels
mergeable $\leftarrow$ true ;
While (mergeable)
mergeable $\leftarrow$ false ;
  Foreach ($q \in K$)
    Foreach ($r \in K / r \neq q \wedge \frac{cover(x,q)|+|cover(x,r)|}{|q \cup r|} \geq \gamma$)
      $K \leftarrow K + q \cup r$;
      $cover(x, q \cup r) = cover(x, q) \cup cover(x, r)$
      mergeable $\leftarrow$ true ;
      toRemove $\leftarrow$ toRemove $+q + r$;
    endFor
  endFor
  Foreach ($k \in$ toRemove) $K \leftarrow K - k$;
  toRemove $\leftarrow \emptyset$
End while
End Algorithm MergeKernels

FIG. 3.12 – L'algorithme MergeKernels

Preuve Soit $k \in K$, un kernel de x en sortie de l'algorithme MERGEKERNELS (*i.e.* k ne peut être fusionné avec aucun autre kernel de K), alors :
1) support$(x, k) > \gamma$. En effet, d'après la définition 15, x est fréquent sur chaque kernel. De plus, si k est le résultat d'une fusion, alors l'algorithme MERGEKERNEL vérifie la fréquence de x sur la période résultante.
2) $\forall q / k \subseteq q$ on a un des cas suivants :

- $x \in k - q$, alors x n'est pas fréquent sur q (sinon, soit k' le kernel dans lequel apparaît x sur q, alors k et k' auraient été fusionnés).
- $x \notin k - q$, alors couverture(x, q)=couverture(x, k) (dans ce cas, x peut rester fréquent ou pas, selon la taille de q).

3) D'après la définition 15, x est supporté par la première et la dernière transaction de k. Alors, x aura une couverture moindre sur toute sous-période de k.
Sur la base des trois observations précédentes, soit $T_x = \{(x, k, \sigma) \forall k \in K)\}$ l'ensemble des itemsets temporels qui correspondent à tous les kernels fusionnés de x sur P avec le support γ, alors T_x est l'ensemble des itemsets compacts $s = (x, x_p, \sigma)$ sur P avec le support minimum γ □

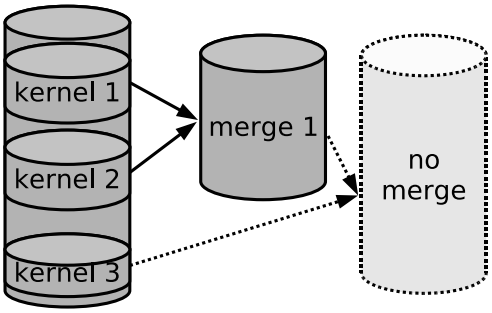

FIG. 3.13 – Fusion des kernels

3.5.2 Algorithme DEICO

Notre algorithme se base sur le principe de la génération de candidats. Soit $c \in C_k$ a candidat de taille k dans l'ensemble des candidats (C_k). Alors, dans notre structure de données, c est associé à $c.i$, l'itemset, $c.p$, la période de fréquence potentielle (*i.e.* les limites dans lesquelles c doit être testé sur les transactions) et $c.kernel$, l'ensemble des kernels de $c.i$ sur $c.p$ avec le support minimum γ (un de nos objectifs est d'extraire $c.kernels$ pour tous les candidats dans C_k au cours d'une passe unique sur les données). De plus, une valeur booléenne permet de connaître l'état du kernel courant ("kernel_fermé" signifie que la définition 15 n'a pas été respectée au cours de la passe). Pour chaque kernel $c.kernel_i$ d'un candidat c, on a $c.kernel_i.s$ (estampille de départ du kernel), $c.kernel_i.e$ (fin du kernel), $c.kernel_i.last$ (dernière apparition de $c.i$ dans le kernel courant), $c.kernel_i.freq$ (la fréquence de $c.i$ sur $[c.kernel_i.s..c.kernel_i.e]$) et $c.kernel_i.cov$ (la taille de la couverture de $c.i$ sur $[c.kernel_i.s..c.kernel_i.e]$). Enfin, $c.current$ représente le kernel courant de x (le dernier kernel ouvert). Considérons C_k, un ensemble de candidats. Pendant la passe sur les données, le but de l'algorithme UPDATE (figure 3.14) est de mettre à jour les informations sur les kernels d'un itemset temporel candidat dont la période englobe l'estampille de la transaction courante. A la fin de chaque passe de l'algorithme SIM (figure 3.15) nous obtenons tous les kernels de chaque candidat pour cette passe. A la fin de chaque passe, les kernels obtenus pour chaque itemset temporel candidat sont fusionnés pour obtenir des itemsets compacts. Le principe de génération de SIM est basé sur le lemme 2.

Lemme 2 *Soit γ le support minimum et x un itemset compact. Alors $\forall i \subset$*

$x_i/|i| = |x_i - 1|, \exists q/x_p \subseteq q \wedge frequence(i,q) \geq \gamma$.

La preuve est immédiate car x est un itemset compact et x_i est fréquent sur x_p. Donc tout sous-ensemble de x_i est fréquent sur x_p.

Nous ne donnons pas les détails sur la génération des candidats de taille 2. Ce cas est similaire au cas de taille supérieure, mais les candidats sont produits par auto-jointure $SI_1 \times SI_1$ et filtrés par l'intersection des périodes. La génération de candidats de l'algorithme GENERATECANDIDATES (figure 3.16) est basée sur les propriétés des lemmes 1 et 2

Théorème 1 *A chaque itération de l'algorithme* DEICO *:* $SI_k \subseteq C_k$ *(i.e.* $\forall s \in SI_k, \exists c \in C_k/s_x = c_i \wedge s_p \subseteq c_p$*).*

Preuve Grâce au lemme 2 on sait que $\forall s \in SI_k, \exists u, v \in SI_{k-1}$ tel que :

1. u_x et v_x sont des préfixes de taille $k-1$ de s.
2. u_x et v_x sont fréquents sur u_p et v_p avec $s_p \subseteq u_p$ et $s_p \subseteq v_p$.
3. u_x n'est pas fréquent sur $v_p - u_p$ (car u est un IC, fréquent uniquement sur u_p, sa période).
4. v_x n'est pas fréquent sur $u_p - v_p$.

Donc, si on étend chaque itemset d'un itemset compact de SI_{k-1} avec tous les items possible et qu'on limite leur période de fréquence potentielle aux intersections qui correspondent au $(k-1)-$itemsets compacts, on obtient un sur-ensemble de SI_k. Il est clair que l'algorithme GENERATECANDIDATES construit ses candidats sur ce principe et limite les périodes de fréquence potentielles à ces intersections. Enfin, grâce au lemme 1 on sait que la détection des kernels de C_k (les $k-$candidats) sur les intersections pertinentes et la fusion de ces kernels, conduisent à la découverte des $k-$itemsets compacts
□

3.5.3 Expérimentations

Les expérimentations sont réalisées sur les fichiers log Web de l'Inria Sophia de Mars 2004 à Juin 2007 qui représentent 253 Go de données brutes et 36 710 616 navigations après le prétraitement.

Comportements découverts. Soulignons d'abord le fait qu'un comportement qui se retrouve dans, par exemple, 15 navigations sur une journée peut-être considéré comme très fréquent. Cela est dû au fait que les caches et

H **Algorithm** UPDATE
In : c, le candidat ; d, la transaction ;
γ, le support minimum
Out : le(s) kernel(s) de c mis à jour
If $(c.kernel\_closed)$
If $(c.i \subseteq d)$ // Démarrer un kernel
$c.current \leftarrow new\_kernel$;
$c.kernel\_closed \leftarrow False$;
$c.current.s \leftarrow d.timestamp$;
$c.current.e \leftarrow d.timestamp$;
$c.current.last \leftarrow d.timestamp$;
End if
Else if $(c.i \subseteq d)$// Continuer le kernel courant
$c.current.e \leftarrow d.timestamp$;
$c.current.last \leftarrow d.timestamp$;
$c.current.cov++$;
$c.current.freq \leftarrow \frac{c.current.cov}{|[c.current.s..c.current.e]|}$;
Else // Vérifier la validité du kernel courant
// *i.e.* $(c.i \not\subseteq d) \Rightarrow$ doit-on fermer le kernel courant ?
$c.current.e \leftarrow d.timestamp$;
$c.current.freq \leftarrow \frac{c.current.cov}{|[c.current.s..c.current.e]|}$;
If $(c.current.freq < \gamma)$
$c.current.e \leftarrow c.current.last$
$c.kernel\_closed \leftarrow True$;
End if
End if
End algorithm UPDATE

FIG. 3.14 – L'algorithme UPDATE

H **Algorithm** SIM
In : γ, le support minimum ; D la base de transactions ;
$\mathcal{I}$ l'ensemble des items
Out : SI l'ensemble des itemsets compacts
qui respectent γ sur D
$k \leftarrow 0$;
Foreach $(i \in \mathcal{I})$
// Construire un candidat correspondant à
// chaque item et l'associer à un ensemble
// vide de noyaux
$C_1 \leftarrow C_1 + (i, [D_s..D_e], \emptyset)$
End for
Do
$k++$;
$SI_k \leftarrow \emptyset$;
Foreach $(d \in D)$; // passe sur les données
Foreach $(c \in C_k / d_{time} \in c.p)$
// L'estampille de d correspond à la période de c
UPDATE(c,d,γ) ;
End for
End for
Foreach $(c \in C_k)$
MERGEKERNELS(c) ;
Foreach $(p \in c.kernels)$
$SI_k \leftarrow SI_k + (c_i, p, frequence(c_i, p))$;
End for
$C_{k+1} \leftarrow$GENERATECANDIDATES(SI_k)
While $(C_{k+1} \neq \emptyset)$
And algorithm SIM

FIG. 3.15 – L'algorithme SIM

Algorithm GenerateCandidates
In : SI_k l'ensemble des itesmsets compacts
de taille k
Out : C_{k+1} l'ensemble des candidats de taille $k+1$
$C_{k+1} \leftarrow \emptyset$
Foreach $x, y \in SI_k$ tels que :
$(x_{i_1}, ..., x_{i_{k-1}}) = (y_{i_1}, ..., y_{i_{k-1}})$
$\wedge y_{i_k} > x_{i_k} \wedge |x_p \cap y_p| > 1$
//les périodes x et y ont une intersection
//non vide et leurs préfixes correspondent
//au critère de génération.
$z = (x_{i_1}, ..., x_{i_{k-1}}, y_k)$
$C_{k+1} \leftarrow C_{k+1} + (z, x_p \cap y_p, \emptyset)$
End for
End algorithm GenerateCandidates

FIG. 3.16 – L'algorithme GenerateCandidates

proxies masquent une grande partie des requêtes sur le site. Mais d'un autre côté, sur nos données de log, un comportement partagé par 15 navigations présente un support de 4.10^{-5} (trois ans d'enregistrement). Notre objectif n'est pas d'extraire des comportement avec un support minimum $\gamma \approx 0\%$ (le chapitre 2 traite de ce sujet). En fait, grâce à la définition d'itemsets compacts, nous pouvons extraire des motifs dont le support est très faible, tout en étant très fréquents sur leurs "régions d'intérêt" avec un temps de calcul optimisé. Voici quelques comportements trouvés grâce aux itemsets compacts, sur le log Web de l'Inria Sophia-Antipolis de 2004 à 2007.

1) Joan Miro : *start* : Thu Apr 20 07 :05 :39 2006 ; *end* : Thu Apr 20 17 :21 :06 2006 ; *frequence* : 0.024565 ; *cover* : 120 ; Préfixe de l'itemiset : "omega/personnel/Christophe.Berthelot/") - *itemset* : {css/style.css, Omega/JoanMiro/joanmiro.html}

Pour interpréter ce comportement, il faut savoir que : 1) cette page est dédiée à Joan Miro (un artiste célèbre) 2) Joan Miro est né le 20 avril 1893 et 3) la page de Christophe est classé cinquième dans les résultats Google avec les mots clés "Joan Miro" (au moment de ces expérimentations). Notre conclusion est donc que pour l'anniversaire de Miro (20 avril), les internautes ont massivement fait des recherches sur l'artiste et sont en partie passés par la page de Christophe. Ce comportement se trouve également en 2004, 2005

et 2007.

2) Conférence MC2QMC2004 : *start* : Mon May 17 09 :22 :41 2004 ; *end* : Mon May 17 12 :49 :26 2004 ; *frequence* : 0.0170512 ; *cover* : 19 ; *itemset* : omega/MC2QMC2004,
omega/MC2QMC2004/monday.html, omega/MC2QMC2004/tuesday.html}

En discutant avec les organisateurs (équipe Omega) de la conférence MC2QMC2004, pour comprendre ce comportement, il apparaît qu'un appel à inscriptions a été lancé à la communauté le 17 mai 2004. Cet appel a été rapidement suivi d'un intérêt plus important pour les pages du programme de la conférence.

3) MedINRIA : *start* : Thu Sep 28 01 :53 :45 2006 ; *end* : Thu Sep 28 04 :27 :25 2006 ; *support* : 0.0148305 ; *cover* : 12 ; *itemset* : asclepios/style.css, asclepios/software/MedINRIA, asclepios/software/MedINRIA/download, asclepios/software/MedINRIA/doc }

Cette fois encore, ce comportement trouve une explication auprès de l'équipe concernée. En effet, Asclepios a annoncé une mise à jour de son logiciel MedINRIA en septembre 2006 et envoyé un message aux utilisateurs le 27 (soir). Il en résulte un téléchargement fréquent du logiciel pendant la nuit.

Caractéristiques des itemsets. Pour étudier les caractéristiques des itemsets compacts, nous avons travaillé sur un seul mois de log (mars 2005, 1 205 754 navigations et 99 262 items). Dans nos résultats, le nombre d'itemsets compacts avec une couverture de 2 est très élevé. C'est pourquoi nous avons ajouté un filtre à DeICo pour ne garder que les itemsets dont la couverture est assez grande. Pour comprendre l'impact de la couverture sur le nombre d'itemsets, nous reportons à la figure 3.17(a) le nombre d'itemsets compacts pour chaque couverture pour cinq supports minimum différents : 5%, 4%, 3%, 2% et 1.5%. Nous pouvons observer, par exemple, que nous trouvons dix itemsets pour une couverture de six et un support minimum de 3%. Les couvertures possibles vont de 4 à 18 mais pour la lisibilité du graphique nous ne reportons que les itemsets d'une sélection de couvertures. Le nombre d'itemsets par couverture va 1 à 89 (support $1,5\%$ et couverture 4). La comparaison entre DeICo et les algorithmes d'extraction d'itemsets existants n'est pas facile dans la mesure où nous travaillons sur des supports très faibles et des périodes spécifiques à découvrir dans les jeux de données. Nous avons donc calculé le support moyen des itemsets compacts par rapport au jeu de données tout entier. Considérons, par exemple, un itemset compact x. Nous proposons de faire une passe sur la base et de trouver le support

global de x_i. Cette opération est appliquée à tous les itemsets compacts et le support global moyen est ainsi calculé puis reporté à la figure 3.17(b). On peut y observer que le support global des itemsets compacts extraits avec $\gamma = 2\%$ est de 0.07%. Ce support global est systématiquement très faible, ce qui est une source d'échec pour les algorithmes traditionnels. En admettant que cette extraction soit toutefois possible, il faut souligner le fait que les périodes ne seraient toujours pas extraites et que les itemsets seraient indépendants de la notion de période (et seraient donc très nombreux).

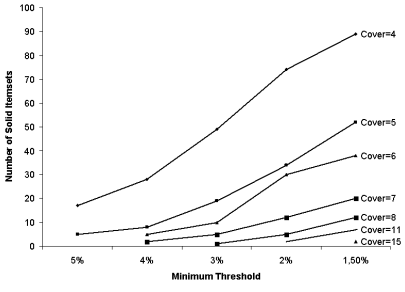

Fig. 3.17(a)

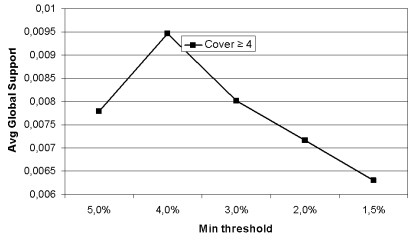

Fig. 3.17(b)

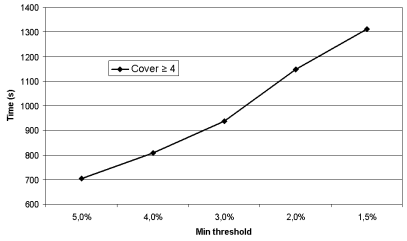

Fig. 3.17(c)

FIG. 3.17 – Caractéristiques des itemsets compacts.

Notre dernière expérimentation est liée aux temps de réponse. La figure 3.17(c) reporte les temps de réponse de DEICO avec différents supports et une couverture minimum de 4. Les temps d'exécution sont encore à améliorer. Toutefois, à l'heure actuelle et à notre connaissance, il n'existe pas de méthode autre que DEICO capable d'extraire ce type de connaissance.

3.6 Discussion

Selon moi, les possibilités de la fouille de données sont parfaitement illustrées par le fameux exemple du motif $\{(Bière,\ Gateaux) \rightarrow (Couches)\ ;\ 10\%\}$. Cet exemple imaginaire d'un motif extrait sur les achats des clients dans un commerce exprime le fait que 10% des clients qui achètent les articles *Bière* et *Gateaux* ont acheté des *Couches*. Ce motif s'expliquerait de la manière suivante : « Madame va faire les courses et de retour elle réalise qu'elle a oublié les couches du bébé. Monsieur va donc au magasin pour les acheter et n'oublie pas, de son côté, d'acheter de la bière et des gateaux ». Cette illustration exprime ce que la fouille de données permet d'extraire : des connaissances auxquelles on ne s'attend par forcément. Sans la fouille de données, le propriétaire du magasin pourrait utiliser son SGBD pour obtenir, par exemple, le nombre de ventes de bières et de gateaux. Mais dans ce cas il n'aurait jamais eu connaissance de ce comportement fréquent. Avec la fouille de données, le propriétaire du magasin n'utilise aucun *a-priori* sur le comportement de ses clients. Il peut « faire parler » les données sans influencer le traitement.

Les travaux présentés dans ce chapitre ont le même objectif. Obtenir des connaissances nouvelles, sans utiliser d'*a-priori* sur ce qui est important (corrélations, fréquences, regroupements) dans les données. En effet, le découpage des données dans lesquelles extraire les connaissances était encore guidé par des idées préconçues des connaissances à extraire. Les travaux présentés dans ce chapitre montrent qu'une considération des données dans leur globalité peut apporter un nouveau type de connaissance : des périodes sur lesquelles les activités sont particulièrement distinctes. Notre approche consiste à reconstruire les différentes périodes (l'historique) qui ont constitué les données. Cependant, le fait de considérer plusieurs mois (ou années) de données, pose un certain nombre de problèmes (plusieurs millions de périodes, trop faible fréquence des comportements, risque d'échec d'un algorithme classique d'extraction de motifs séquentiels sur une de ces périodes, etc.).

Dans un premier temps, nos travaux ont montré qu'avec une heuristique indexant le log période après période, nous pouvions extraire les comportements fréquents (quand il y en avait). Ces comportements ont la particularité d'être ponctuels ou répétitifs, en majorité rares et surtout représentatifs de périodes denses. Nos expérimentations ont permis d'extraire, entre autres, des comportements relatifs à des séances de TP ou encore des navigations sur les pages d'une conférence dans les jours qui précedent la date de soumission.

Dans un deuxième temps, nous avons proposé une nouvelle définition pour la découverte d'itemsets qui correspondent à des fréquences élevées sur des périodes précises sans connaissance préalable sur ces périodes. Cette découverte posait la difficulté de découvrir en même temps les itemsets et leurs périodes optimales de fréquence. De plus, le nombre de combinaisons possible devait être réduit et nous avons apporté les bases théoriques nécessaires à la résolution du problème. Notre algorithme, basé sur la découverte de 'kernels' et leurs fusions, s'est révélé efficace et capable d'extraire ce nouveau type de connaissance de manière précise et exhaustive. D'après nos expérimentations, les itemsets compacts constituent un résultat lisible et instructif pour mieux comprendre les données étudiées.

3.6.1 Extraction de tendances

J'ai également co-dirigé le post-doc de Céline Fiot sur un sujet lié au thème de l'évolution en tant que connaissance. Il s'agissait d'extraire des motifs exprimant des tendances. Par exemple le fait que « dans 50% des cas, le passage à la vitesse supérieure est suivi d'une augmentation rapide du régime du moteur ». L'intérêt de ce type de motif vient de :

- L'expression d'une tendance (« augmentation rapide » du régime du moteur). Cet aspect est abordé grâce à la logique floue.
- Sa générécité (le motif se vérifie quand on passe de la 1^{ere} à la 2^{nde} mais aussi de la 2^{nde} à la 3^{eme}, etc.).

Ces travaux sont décrits dans [FMLT08].

3.6.2 Les flux de données : royaume de l'évolution

Les tavaux présentés dans ce chapitre ont permis de mettre en avant l'existance de périodes pendant lesquelles des motifs étaient fréquents (ces mêmes motifs étant non fréquents sur la globalité des données). Une des raisons de cet aspect évolutif des connaissances vient de l'évolution des usages. Comme je l'indiquais dans l'introduction de ce mémoire, le dynamisme des usages est souvent lié au dynamisme des systèmes (l'exemple typique sur le site Web de l'Inria Sophia étant la conférence organisée par une équipe et qui génère de nombreux usages aux dates d'activités de cette conférence). Quand les systèmes deviennent très dynamiques et qu'ils sont consultés massivement, il n'est plus possible d'analyser ces usages avec les méthodes traditionnelles en raison des volumes de données à analyser et de leur rapidité de production. Il n'est parfois même plus possible de stocker les traces d'usages.

Ces données sont connues sous le nom de flux de données (ou data streams). Le chapitre suivant propose deux études sur ce sujet.

Chapitre 4

Flux de données : l'équilibre entre vitesse et précision pour des données fortement évolutives

Les flux de données produisent des données en quantité potentiellement illimitée, à une vitesse et dans des quantités telles que les outils et méthodes actuels ne permettent pas de les traiter dans leur ensemble. Parmi les conséquences de ces caractéristiques hors-normes, citons :

- La difficulté ou l'impossibilité de stocker les données générées par le flux.
- L'impossibilité d'analyser les données du flux avec les techniques de fouille traditionnelles (la quantité de données rendrait le temps de calcul trop grand).
- L'interdiction de bloquer le flux. En effet, le flux contient souvent les signes vitaux du système qui le produit. Malheureusement les opérations de base de la fouille de données (comme l'auto-jointure entre les items fréquents par exemple) sont typiquement bloquantes.
- La difficulté ou l'impossibilité de stocker les connaissances extraites (pour les mêmes raisons que les données, à long terme leur quantité devient trop importante pour les stocker). Il faut donc historiser ces connaissances en utilisant un principe de compression avec perte.

Dans le domaine de l'extraction de connaissances dans les flux, l'approximation a rapidement été reconnue comme un facteur clé pour fournir des motifs à la vitesse imposée par l'application [GGR02]. Ensuite, des méthodes ré-

centes ([CDH+na, GHP+03, TCY03]) ont introduit différents principes pour gérer l'historique des motifs extraits. L'idée principale est que l'on est généralement plus interessés par les changements récents que par les changements plus anciens. [GHP+03] a ainsi introduit la notion de *logarithmic tilted time window* pour stocker les fréquences des motifs avec une granularité fine pour les changements récents et une granularité plus large pour les changements plus anciens. Dans [TCY03], une technique de regression est utilisée pour représenter les fréquences et une technique permettant de régler la finesse de représentation est introduite. Enfin, dans [RPT05], les auteurs proposent une nouvelle structure de données destinée à extraire les motifs séquentiels fréquents d'un data stream. Dans ce chapitre, nous montrons que les phénomènes combinatoires liés à l'extraction des motifs séquentiels rendent toute méthode exhaustive potentiellement bloquante. En effet, si dans le cas des règles d'association le nombre de possibilité est fini, ce n'est pas le cas des motifs séquentiels, pour lesquels un item peut se répéter à l'infini.

La section 4.5 décrit deux algorithmes d'extraction de **motifs séquentiels approximatifs dans les flux de données** proposés lors de la thèse d'Alice Marascu. Chacun de ces algorithmes est basé sur l'alignement de séquences (comme [KPWD03, HWV02] l'ont déjà utilisée pour la fouille de bases de données statiques). Nos approches satisfont les contraintes liés à la rapidité du data stream et peuvent être incluses dans un environnement temps réel.

La section 4.6 décrit **une nouvelle stratégie de gestion et de résumé de l'historique des motifs extraits**. Ces historiques peuvent être vus comme des séries temporelles. Notre objectif et de proposer une méthode capable (1) de se restreindre à l'espace mémoire disponible et (2) de calculer une approximation efficace et fiable d'un série temporelle. Notre proposition se situe donc à la fois sur une stratégie d'optimisation de l'erreur globale des résumés de plusieurs séries et sur une méthode rapide et fiable de compression des séries. Produire des résumés de séries temporelles, permettant de répondre à ce problème, est un sujet très étudié ces dernières années [CL03, CDH+na, CS03, PSF05, TCY03, ZS02]. Considérons M, la quantité de mémoire disponible. Imaginons que nous souhaitions résumer le comportement de n séries temporelles et considérons que nous disposons d'une mémoire organisée sous la forme d'une matrice à n lignes. Nous pouvons alors stocker en mémoire au maximum $t = M/n$ valeurs pour chaque série. Après t itérations du flux la mémoire est pleine et nous devons appliquer une méthode pour résumer les séries et quand t devient très grand ce résumé ne peut pas se faire sans introduire un degré d'approximation. Ce degré d'approximation se traduit alors par une erreur locale à chaque série et

une erreur globale (cumul des erreurs locales) du modèle. Les questions qui se posent sont alors les suivantes : "quelle série doit être résumée pour minimiser l'erreur globale ?", "Comment résumer une série pour minimiser l'erreur locale ?", "quel algorithme utiliser pour calculer l'approximation aussi vite que possible ?".

Exemple 5 *De nombreux sites Web souhaitent mesurer la fréquence des requêtes sur leurs URLs et la variation de cette fréquence dans le temps. Ces données ne peuvent toutefois pas être stockées étant donnés leur volume et la durée d'observation. Ainsi, un résumé fiable de ces données doit être calculé. Ce résumé doit permettre de montrer, pour chaque URLs observée, la fréquence des requêtes et son évolution dans le temps avec une qualité d'approximation aussi bonne que possible.*

Il existe de nombreuses techniques capables de répondre au problème de l'exemple 5. Une première classe de techniques consiste à minimiser l'erreur locale. Plus précisément, quand la mémoire allouée à une séquence est saturée, cette série doit être compressée. Disons que cette séquence contient plus de t segments (observations) alors une compression de deux (ou plusieurs) segments doit être appliquée. Une autre classe de techniques utilise un facteur d'ancienneté des données. Dans ce cas, les segments les plus anciens sont considérés comme moins importants et, en conséquence, doivent être choisis pour la compression plus souvent que les segments récents. La majorité des solutions existantes dans les flux de données est basée sur la deuxième classe de méthodes (utilisant l'ancienneté). L'idée principale est inspirée de la mémoire humaine qui privilégierait les événements récents. Notre approche se base plutôt sur l'idée que la mémoire humaine peut être plus influencée par les événements marquants que par les événements anodins. Nous proposons la méthode REGLO (Résumés à Erreur Globale Optimisée), une approche rapide pour résumer des flux de séries temporelles en ligne. *L'avantage principal de REGLO est sa capacité à trouver la meilleure approximation possible en fonction de l'erreur globale.* De plus, REGLO fonctionne en temps réel.

Concernant l'approximation d'une série temporelle, de nombreuses techniques ont été proposées dans la littérature. Parmi ces techniques, citons les ondelettes, la transformée de Fourier ou encore la régression linéaire. Le rôle de la technique d'approximation est crucial dans la mesure où cette représentation est mise à jour de manière massive pour résumer le flux. La régression linéaire (RL) consiste à proposer une droite qui approche au mieux (au sens des moindres carrés) les valeurs de la série d'origine. Dans ce cas, l'erreur quadratique est utilisé comme critère de distance entre le modèle obtenu et

les valeurs d'origine. L'erreur quadratique (ou somme des erreurs carrées) est calculée comme la somme des distances carrées entre chaque valeur sur la droite de régression et la valeur d'origine de la série temporelle.

Nous proposons également *une nouvelle technique d'approximation*. Avec la RL, la fusion de deux segments peut être calculée en O(1) (ainsi que l'erreur résiduelle). Toutefois, les articles existants dans le domaine sont basés sur des formules complexes impliquant des variables et des opérateurs nombreux. Nous proposons tout d'abord de simplifier ces formules. Ensuite, compte tenu de la vitesse à laquelle les flux doivent être traités, nous proposons une technique d'approximation rapide qui donne des résultats similaires à ceux de la RL tout en utilisant beaucoup moins de variables et d'opérateurs. AMi (Approximation par les Milieux), notre nouvelle technique d'approximation, utilise les milieux des segments pour les résumer.

Les contributions, présentées dans les sections 4.5 et 4.6, sont présentées plus en détails dans [Mar09].

4.1 Activités

Je suis porteur du projet SéSur (ARC Inria, 2006-2008, budget 24KEuros) sur le thème des flux de données et de la sécurité. Ce projet a donné lieu a des collaborations avec le LIRMM, le LGI2P et l'IRISA.

Je suis également responsable, pour le partenaire Inria, du projet ANR MIDAS (Mining Data Streams) sur ce thème (part de l'Inria : 91KEuros).

Enfin, ce travail fait l'objet d'un transfert technologique (en cours de signature) vers Orange Labs via un CRE (Contrat de Recherche Externalisée) d'un montant de 40KEuros pour l'Inria.

4.1.1 Encadrement

- **Doctorante** : j'ai participé à l'encadrement d'Alice Marascu (sous la direction d'Yves Lechevallier, Inria). Alice a soutenu sa thèse le 14 septembre 2009 : « Extraction de motifs séquentiels dans les flux de données ». Les travaux correspondants sont décrits dans les sections 4.5 et 4.6. Taux de participation à l'encadrement d'Alice Marascu : 75%. Alice effectue à ce jour un post-doc à l'IRISA.

4.1.2 Publications

Les travaux décrits dans ce chapitre ont donné lieu à des publications dans deux revues internationales, dans deux conférences nationales et dans

deux ateliers internationaux :

- Alice Marascu and Florent Masseglia. « Atypicity Detection in Data Streams : a Self-Adjusting Approach ». In Intelligent Data Analysis Journal (IDA). To appear. 2009.
- Alice Marascu and Florent Masseglia. « Mining Sequential Patterns from Data Streams : a Centroid Approach ». In Journal for Intelligent Information Systems (JIIS). Issue 27, Number 3, pp 291-307. November 2006.
- Alice Marascu, Florent Masseglia. « Classification de flots de séquences basée sur une approche centroïde », INFORSID Informatique des organisations et systèmes d'information et de décision, Hammamet, Tunisie, June 2006.
- Alice Marascu, Florent Masseglia. « Extraction de motifs séquentiels dans les flots de données d'usage du Web », Extraction et Gestion des Connaissances (EGC'06), Lille, January 2006.
- A. Marascu and F. Masseglia. « Mining Data Streams for Frequent Sequences Extraction ». In IEEE first Workshop on Mining Complex Data (MCD'05). Held in conjunction with ICDM'05, Houston, USA, November 27, 2005.
- A. Marascu and F. Masseglia. « Mining Sequential Patterns from Temporal Streaming Data ». In ECML/PKDD first Workshop on Mining Spatio-Temporal Data (MSTD'05). Held in conjunction with PKDD'05, Porto, Portugal, October 3-7, 2005.

4.2 Définitions

4.2.1 Flux de données

Il n'existe pas de définition universellement admise d'un flux de données. Généralement, il s'agit des mêmes formats de données que ceux traités dans les cas statiques, avec pour contraintes supplémentaires :

- une grande vitesse de production ;
- de grande quantités de production ;
- l'interdiction de bloquer le flux ;
- l'impossibilité de stocker les données du flux ;
- l'impossibilité d'appliquer les méthodes conçues pour les données traditionnelles sans bloquer le flux.

Considérons la définition 2 de la section 2.2. Une définition d'un flux, dans le cas de ces données, pourrait être la suivante :

Définition 16 Soit S un ensemble de séquences (au sens de la définition 2). Alors $S = \{s_1 ... s_n\}$ avec s_i une séquence de taille m telle que $s_i =< s_{i_1} \dots s_{i_m} >$, et s_{i_1} l'itemset observé dans s_i au temps 1. Soit une fenêtre d'observation F_x^y contenant un sous-ensemble de S sur la période $[x \dots y]$, avec $y > x$. Alors $F_x^y = \{\forall s_k \in S, < s_{k_o} \dots s_{k_p} > | o > x \text{ et } p < y\}$. Un flux DS est représenté par l'ensemble des fenêtres F_x^y pour tout x et tout y définis par l'utilisateur avec des valeurs strictement croissantes dans le temps.

Il existe différentes versions de fenêtres pour observer un flux et plusieurs problèmes d'extraction à résoudre dans ce contexte. J'en propose un tour d'horizon dans la section 4.3

4.2.2 Résumés optimisant l'erreur globale

Une fois les motifs extraits d'un flux, il faut en gérer l'historique. Nous considérons l'historique de la fréquence des motifs comme des séries de valeurs numériques (une série par motif). Nous proposons alors le problème qui consiste à résumer un tel ensemble de séries de la manière suivante :

Soit $T[1..n]$, un ensemble de n séries temporelles à observer. Soit $T[j]$ la j^{eme} série de T alors à l'étape s on a $T[j] = (T[j][1], ..., T[j][s]]$ comme ensemble des observations (valeurs) de $T[j]$. Soit M la quantité de mémoire disponible (en d'autres termes, on peut stocker M segments ou valeurs en mémoire). Nous considérons le cas de temps discrétisé. A chaque étape nous avons une valeur pour chaque série (la valeur par défaut étant zéro). La valeur maximum de s permettant de stocker les mesures sans compression est $s = M/n$. Quand $(s \times n) > M$, la représentation des séries doit être compressée et la perte d'information est liée à la différence entre ces deux nombres. Une technique de compression de séries temporelles bien connue se trouve dans la régression linéaire (RL). Soit $R[1..n]$, l'ensemble des représentations des n séries et $S[i]$ la taille de $R[i]$ (*i.e.* le nombre de segments utilisés par la représentation $R[i]$). La représentation $R[i]$ de la série i est une approximation de cette série en $S[i]$ segments avec comme contrainte globale $\sum_{i=1}^{n} S[i] = M$. Soit $E[i]$, l'erreur de $R[i]$ par rapport aux données d'origine de la série i et $E(R)$, l'erreur globale de $R[1..n]$. Alors $E(R) = \sum_{i=1}^{n} E[i]$.

Les travaux existants se sont concentrés sur :

1. L'optimisation de l'erreur d'une seule représentation par la RL (*i.e.* optimiser $E[i]$ en fusionnant les segments de $R[i]$).
2. Ou bien sur l'importance accordée aux événements récents.

L'approximation obtenue par ces méthodes est telle que $\forall i, j \in [1..n]$, $S[i] = S[j]$ (la taille des représentations est identique d'une série à l'autre). À notre connaissance, il n'existe pas de travaux pour résumer un ensemble de séries temporelles en optimisant l'erreur globale (*c.à.d* optimisant $E(R)$ en fusionnant et en allouant les segments de R à chaque étape s). *Le premier problème traité dans la section 4.6 porte sur la recherche d'une distribution optimale des M segments disponibles afin de minimiser l'erreur globale $E(R)$ dans un traitement en ligne.*

4.2.3 Fusion des segments

Comme nous l'expliquons en section 4.6.2, la RL appliquée sur deux segments peut-être calculée en O(1). En nous basant sur la RL, nous proposons une formule impliquant un nombre considérablement réduit de variables et d'opérateurs. *De plus, nous réduisons le nombre d'opérations nécessaires à la fusion de deux segments.* Notre proposition pour une technique rapide d'approximation est expliquée en section 4.6.3.

4.3 Travaux existants

Un flux produit de nouvelles données en continu et n'est pas supposé terminer. Travailler sur l'ensemble du flux en permanence, avec un niveau de détails élevé, conduirait à des opérations bloquantes sur le flux, ce qui est contraire aux spécifications de son analyse. Extraire des motifs (items, itemsets ou motifs séquentiels) dans un flux de données, impose donc de gérer deux facteurs afin de trouver le meilleur équilibre entre les temps de traitement et le niveau de détails de l'analyse. Le premier facteur s'exprime sous la forme d'une fenêtre qui correspond à la vue que l'utilisateur veut explorer dans le flux. Cette fenêtre peut correspondre à la vue qui est accordée sur les données, ou bien à la façon de gérer l'historique des connaissances extraites. Si cette fenêtre recouvre l'ensemble du flux avec un niveau de détail élevé, alors on se retrouve dans le cas bloquant évoqué plus haut.

Dans cette section, je propose une étude de la gestion des fenêtres temporelles dans les méthodes d'extraction de motifs sur les flux de données. Cette section est organisée de la manière suivante. Dans un premier temps, en section 4.3.1, j'aborde l'extraction de motifs fréquents sur un flux de données en adoptant le point de vue de la gestion des fenêtres de temps utilisées pour cette extraction. Dans la section 4.3.2, je propose un tour d'horizon d'un problème récent sur les flux : l'extraction de rafales. En effet, ce problème

impose de travailler avec des définitions précises des fenêtres d'observation d'un flux.

4.3.1 Extraction de motifs et fenêtres temporelles

Les méthodes d'extraction de motifs actuelles peuvent être classées dans trois catégories, selon la gestion qu'elles ont de cette fenêtre et de sa taille. La première catégorie correspond aux fenêtres "avec drapeau de début" (ou *landmark*). Il s'agit de fenêtres qui commencent à une date précise et terminent à la date la plus récente du flux (date actuelle). Les méthodes de la deuxième catégorie utilisent des fenêtres de tailles variables (*tilted time windows*, *decay rate* ou encore *time fading models* sont les mots clés employés pour ces fenêtres) selon leur emplacement dans l'écoulement du flux. Cette taille est croissante avec le temps, de sorte que les événements les plus récents sont observés dans des fenêtres plus petites (ils sont donc moins nombreux dans une fenêtre) et les événements les plus anciens sont observés dans des fenêtres plus grandes (ils sont donc plus nombreux dans une fenêtre). Ces méthodes sont basées sur l'idée que les événéments les plus récents sont les plus intéressants. La troisième catégorie correspond aux fenêtres glissantes (*sliding windows*). Dans ce cas, les fenêtres ont une taille fixe W et seules les W dernières transactions sont prises en compte lors de la fouille. Enfin, le deuxième facteur (le niveau de détail) dépend de la disponibilité obtenue auprès du système une fois que la taille de la fenêtre et sa méthode de gestion ont été définies.

Fenêtres avec drapeau de début

Dans les méthodes à drapeau de début, les données utilisées se situent entre le drapeau et la date la plus récente du flux. En général, le drapeau de début correspond à l'initialisation du système. Dans la mesure où ces méthodes ne gèrent pas différentes fenêtres de différentes tailles, je ne donne pas de détails techniques sur les algorithmes présentés. En effet, ces méthodes partagent un point commun : la taille de la structure augmente avec le déroulement du flux et l'approximation n'est pas calculée en fonction de l'aspect récent ou ancien des enregistrements. La figure 4.1 illustre ce type d'approches. On peut y constater que la gestion de la fenêtre n'est pas le problème principal de ces approches. Il s'agit principalement de faire grandir cette fenêtre en même temps que le flux est alimenté en nouvelles données.

Dans [MM] on trouve un algorithme permettant de déterminer de façon approximative la fréquence des éléments d'un flux de données. L'algorithme

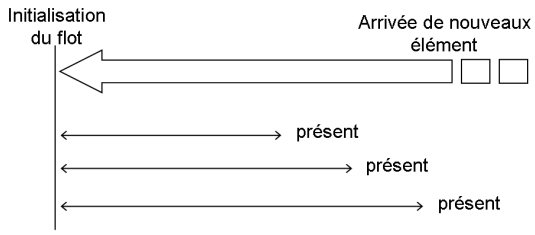

FIG. 4.1 – Fenêtres à drapeau de début (landmark)

(Lossy-counting) utilise tout l'historique du flux pour calculer la fréquence des motifs de façon incrémentale. Lossy-counting est représentatif des approches à drapeau. Le flux est divisé en une séquence de paquets. Chaque paquet est également identifié par un ID. L'algorithme maintient un ensemble de variables afin de suivre le nombre maximum d'occurences possibles de chaque itemset dans les données passées. L'ensemble $\mathcal{D}$ de variables maintenues est de la forme $(X, freq(X), err(X))$ où X est un itemset fréquent, $freq(X)$ sa fréquence depuis son insertion dans $\mathcal{D}$ et $err(X)$ une limite supérieure à la fréquence de X avant son insertion dans $\mathcal{D}$. Ensuite, selon un paramètre de tolérance d'erreur et un support minimum, cette approche met à jour le compteur de chaque itemset dont le comptage approximatif dépasse le support minimum. Tous les itemsets fréquents sont extraits par lossy-counting (il n'y a pas de faux négatif). Cependant cette méthode travaillant sur l'ensemble du flux, selon le modèle des méthodes *landmark*, présente des compteurs qui augmentent avec le temps et ne peuvent pas, à terme, rester en mémoire centrale.

Dans [HLS] les auteurs proposent la structure de données SFI-forest (summary frequent itemsets forest) afin de maintenir l'ensemble des itemsets fréquents trouvés dans le flux depuis son initialisation. L'algorithme proposé (DSM-FI) repose sur le même calcul d'erreur que [MM]. La structure SFI-forest est une extension des structures de résumé basées sur les arbres préfixés. A l'arrivée d'une nouvelle transaction T contenant m items, T est projetée en m sous-transactions qui sont insérées dans la structure selon leurs suffixes. DSM-FI utilise un support étendu ϵ afin de contrôler la précision des résultats de la fouille. Périodiquement, DSM-FI supprime les itemsets qui ont un support inférieur à ϵ. L'arbre proposé par DSM-FI est plus compact qu'un arbre préfixé. Par exemple l'itemset $x_1x_2x_3$ est représenté par un seul chemin $< x_1, x_2, x_3 >$ dans l'arbre suffixé mais sera représenté par

deux chemins ($< x_1, x_2, x_3 >$ et $< x_1, x_3 >$) dans un arbre préfixé. Toutefois cette compacité se paie lors des calculs qui seront plus nombreux sur l'arbre suffixé, dans la mesure ou des parcours d'arbre nombreux seront nécessaires pour déterminer la fréquence des itemsets.

Dans [JA05], les auteurs proposent un algorithme basé sur une seule passe pour découvrir les motifs fréquents d'un ensemble de données. Leur algorithme prend en entrée un support minimum σ et une variable d'erreur ϵ. L'algorithme extrait alors tous les itemsets dont la fréquence est supérieure à σ et n'extrait pas les itemsets sont la fréquence est inférieure à $(1 - \epsilon)\sigma$. Durant l'extraction, l'occupation en mémoire augmente proportionellement à $1/\epsilon$. Les auteurs veulent garantir deux fontionnalités de leur algorithme :

- Calculer les k-itemsets de manière approximative lors de la première passe en conservant une structure de données de taille faible.
- Proposer une borne supérieure à la précision des résultats après la première passe, afin d'éviter une seconde passe (impossible sur un flux de données). En d'autres termes, éviter d'insérer des faux positifs dans le résultat du point précédent.

Les auteurs proposent une structure de données basée sur un arbre de hachage préfixé.

Fenêtres en pente

Les méthodes à drapeau de début sont limitées par le fait qu'elles accordent à tous les événements la même importance. Il n'y a pas de distinction entre les enregistrements et donc la mémoire est distribuée de manière uniforme pour tout l'historique du flux. Une piste de recherche pour pondérer la quantité de mémoire affectée aux données du flux se trouve dans les fenêtres "en pente". Plusieurs termes sont employés pour désigner ces techniques : *tilted time windows*, *decay rate* ou encore *time fading models*. L'idée générale est que l'on est plus intéressé par les événements récents que par les anciens. Ainsi, les propositions faites dans le domaine des fenêtres penchées consistent à gérer l'historique des connaissances extraites avec une granularité plus fine pour les événements récents. Cette granularité devient plus grossière avec le temps. La figure 4.2 illustre cette gestion. Pour un événement correspondant au paquet de données n on voit que la gestion se fait d'abord avec la granularité la plus fine (le paquet n se voit attribué une fenêtre complète pour gérer l'historique des connaissance qui y sont extraites). Ensuite, au fil des mises à jour, l'historique du paquet n est confondu avec celui des paquets précédents, jusqu'à la granularité la plus large de l'historique, quand le paquet n est devenu très ancien.

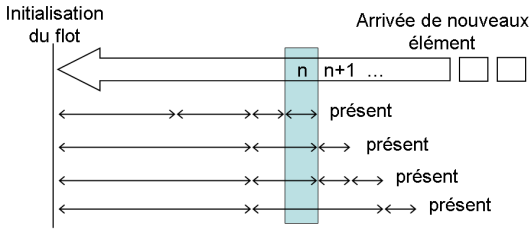

FIG. 4.2 – Fenêtres en pente (decay, tilted, time-fading)

Un facteur de vieillissement (*decay rate*) est proposé dans [CL03]. J'explique dans la suite la façon dont ce facteur d est appliqué à l'arrivée de chaque transaction.

Considérons D_k, l'ensemble des transactions (composées d'items) arrivées dans le flux jusqu'au temps k, $D_k = \{T_1, T_2, ..., T_k\}$. Sans facteur de vieillissement, la taille du flux (nombre de transactions) serait donnée par le nombre de transactions. Les auteurs de [CL03] proposent d'appliquer en permanence un facteur de vieillissement qui biaise le calcul sur la taille du flux. Ce facteur d, tel que $0 < d < 1$, est utilisé de la manière suivante :

$$|D| = \begin{cases} 1 & \text{si } k = 1 \\ |D|_{k-1} \times d + 1 & \text{si } k \geq 2 \end{cases}$$

Les auteurs prouvent que $|D|$, en appliquant le facteur d, converge vers $1/(1-d)$ si k tend vers l'infini. Le calcul de fréquence d'un itemset X à l'arrivée de la transaction k (*i.e.* $Freq_k(X)$) est alors également biaisé par ce facteur de la manière suivante (version simplifiée) :

$Freq_k(X) = d^{k-1} \times w_1(X) + d^{k-2} \times w_2(X) + ... + d^1 \times w_{k-1}(X) + w_k(X)$.

Où d^n correspond à n applications du facteur d à lui-même et :

$$w_i(X) = \begin{cases} 1 & \text{si } X \in Y_i \text{ (la transaction } Y_i \text{ supporte } X) \\ 0 & \text{sinon} \end{cases}$$

L'algorithme [TCY03] propose de gérer deux types de fenêtres lors de l'extraction des itemsets fréquents sur un flux de données. La première fenêtre est ralative à l'extraction des motifs. C'est une fenêtre glissante et les détails sont donnés dans la section 4.3.1. La deuxième fenêtre est une fenêtre en pente, destinée à gérer l'historique des motifs extraits. Considérons l'historique dont la gestion est illustrée par la figure 4.3. Les auteurs proposent

d'utiliser un modèle de PLR (*Piecewise Linear Representation*) qui est utilise une technique de segmentation d'une série temporelle (l'historique pouvant être considéré comme une telle série). Dans ce modèle, un seuil est défini par l'utilisateur afin de déterminer l'erreur maximum pour la représentation. Les auteurs proposent donc d'ajouter la notion d'ajustement pour les segment de la PLR obtenue sur un historique. Toujours en se basant sur l'idée que les utilisateurs peuvent être plus intéressés par les événements récents plutôt que par les anciens, les auteurs proposent d'ajuster la granularité de la représentation en fonction de l'intérêt porté sur une période.

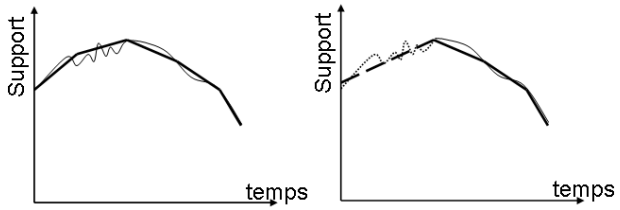

FIG. 4.3 – Technique d'ajustement pour les segments anciens

Toutes ces approches proposent une approximation dans leur extraction de connaissance et également dans la gestion de l'historique de ces connaissances. Elles ajustent la précision de l'historique enregistré en fonction de la mémoire disponible.

Fenêtres glissantes

Les méthodes précédentes proposent une gestion de l'historique qui couvre toute l'existance du flux, en pondérant le niveau de détail par l'age des données. Dans certaines applications, les utilisateurs peuvent n'être intéressés que par les événements les plus récent (et uniquement ceux-là). Les modèles de fenêtres précédents ne sont pas en mesure de satisfaire ce point de vue. Les modèles de fenêtres glissantes, illustrées par la figure 4.4 répondent à cette demande [CL04, CWYM04, AM04, BDMO03, JCLR07, WF06]. Étant donnée une fenêtre de taille W, seules les W dernières transactions sont utilisées pour la fouille. Quand une nouvelle transaction arrive, les transactions les plus anciennes expirent. Ce modèle nécessite donc des méthodes capable de mettre à jour le support des motifs en fonction de l'expiration et de l'ajout de transactions.

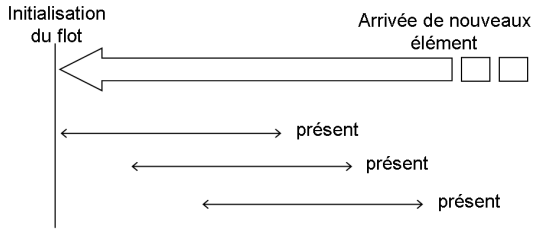

FIG. 4.4 – Fenêtres glissantes (sliding)

[BDMO03], par exemple, présente un mécanisme qui peut combiner de manière dynamique les paquets de données dans un histogramme afin d'observer la variance dans une fenêtre glissante. Leur solution au problème de la mise à jour de cette variance repose sur une estimation continuellement mise à jour de la variance pour les N dernières valeurs dans le flux avec une erreur relative d'au moins ϵ.

L'algorithme *estwin* [CL05] propose de maintenir les itemsets fréquents sur une fenêtre glissante. Les itemsets générés par *estwin* sont maintenus dans un arbre préfixé $\mathcal{D}$. Un itemset X dans $\mathcal{D}$ est composé de trois champs : $freq(X)$, $err(X)$ et $tid(X)$ avec $freq(X)$ la fréquence de X sur la fenêtre actuelle, $err(X)$ la borne supérieure de l'erreur sur la fréquence de X dans la fenêtre actuelle avant que X soit inséré dans $\mathcal{D}$ et $tid(X)$ l'ID de la transaction en cours de traitement. Pour chaque transaction entrante Y avec un ID tid_τ, *estwin* incrémente la fréquence de chaque sous-ensemble de Y dans $\mathcal{D}$. Soit N le nombre de transactions dans la fenêtre et tid_1 l'ID de la première transaction, *estwin* supprime l'itemset X et tous ses sur-ensembles si une des deux conditions suivantes est respectée :

1. $tid(X) \leq tid_1$ et $freq(X) < \lceil \epsilon N \rceil$
2. $tid(X) > tid_1$ et $freq(X) < \lceil \epsilon(N - (tid(X) - tid_1)) \rceil$.

$tid(X) < tid_1$ signifie que X est arrivé dans (D) avant la fenêtre actuelle, car tid_1 est l'ID de la première transaction dans la fenêtre actuelle. Si $tid(X) > tid_1$ alors X est arrivé dans $\mathcal{D}$ dans la fenêtre actuelle et $(N-(tid(X)-tid_1))$ donne le nombre de transactions arrivées dans la fenêtre actuelle après la transactions d'ID $tid(X)$.

Après avoir mis à jour et supprimé les itemsets concernés, *estwin* procède à l'insertion des nouveaux itemsets dans (D). Premièrement, les nouveaux items sont ajoutés à $\mathcal{D}$. Ensuite, pour tout itemset X tel que $|X| \geq 2$ si tous

les sous-ensemble de X sont déjà dans $\mathcal{D}$ avant l'arrivée de la transaction Y alors *estwin* ajoute X dans $\mathcal{D}$ puis met à jour l'erreur estimée sur X.

Les auteurs de [CL04] proposent une méthode inspirée par le mécanisme de l'algorithme lossy-counting [MM] pour maintenir les itemsets fréquents sur une fenêtre glissante. La structure d'arbre préfixé $\mathcal{D}$ de l'algorithme *estwin* est conservée pour maintenir la fréquence des itemsets. Un itemset X de $\mathcal{D}$ garde l'information $freq(X)$, la fréquence de X dans la fenêtre actuelle au moment de l'insertion de X dans $\mathcal{D}$ et $tid(X)$ qui correspond à l'ID de la transaction en cours de traitement au moment de cette insertion. Pour chaque nouvelle transaction Y d'ID tid_τ, l'algorithme incrémente la fréquence de chaque sous-ensemble de Y dans $\mathcal{D}$ et y insère tout nouvel itemset X tel que $X \subseteq Y$ avec $freq(X) = 1$ et $tid(X) = tid_\tau$. Pour chaque sous-ensemble X d'une transaction sortante, si $X \in \mathcal{D}$, si $tid(X) \leq tid_1$ alors une unité de valeur est supprimé à $freq(X)$. L'algorithme supprime ensuite X et ses sous-ensembles si une des deux conditions est respectée :

1. $tid(X) \leq tid_1$ et $freq(X) < \lceil \epsilon N \rceil$
2. $tid(X) > tid_1$ et $(freq(X) + \lfloor \epsilon(tid(X) - tid_1) \rfloor) < \lceil \epsilon N \rceil$.

Dans [CWYM04] les auteurs proposent d'extraire des itemsets fréquents fermés sur un flux de données. L'algorithme *Moment* met à jour de manière incrémentale l'ensemble des itemsets clos fréquents sur une fenêtre glissante. Les auteurs proposent une structure basée sur un arbre préfixé : le CET (*Closed Enumeration Tree*. soit v_x un nœud représentant un itemset X dans le CET. v_x peut être classé dans un des groupes suivants :

- IGN (*Infrequent Gateway Nodes*) : v_x est un IGN si X n'est pas fréquent, v_y est le père de X et Y est fréquent et si v_y a un frêre $v_{y'}$ tel que $X = X \cup Y$ alors Y' est fréquent.
- UGN (*Unpromising Gateway Nodes*) : v_x est un UGN si X est frequent et $\exists Y$ tel que Y est un frequent clos, $Y \supset X$, $freq(Y) = freq(X)$ et Y est avant X dans l'ordre lexicographique des itemsets.
- IN (*Intermediate Node*) : v_x est un IN si X est fréquent, v_x est le père de v_y tel que $freq(Y) = freq(X)$ et v_x n'est pas un UGN.
- CN (*Closed Node*) : v_x est un nœud fermé si X est un frequent fermé.

A l'arrivée d'une nouvelle transaction, comme à l'expiration d'une transaction existante, *Moment* parcours l'arbre CET afin de mettre à jour les nœud et leur catégorie ou bien dans le but d'ajouter et supprimer des nœuds. L'initialisation se fait en déterminant les IGN grâce à la propriété d'antimonotonie des itemsets fréquents.

[TCY03] propose l'algorithme FTP-DS, destiné à extraire des itemsets

dans un flux de données. FTP-DS utilise en réalité deux types de fenêtres. Le premier type de fenêtre, destiné à extraire les itemsets, est une fenêtre glissante. Le second type de fenêtre est une fenêtre en pente destinée à gérer l'historique des motifs extraits, comme expliquée dans la sous-section 4.3.1. Afin d'extraire les motifs, FTP-DS propose d'adapter le principe d'Apriori (Générér-Elaguer) en utilisant un passe d'Apriori à chaque mise à jour de la fenêtre. Pour pousser encore plus loin l'analogie avec Apriori, on peut imaginer l'algorithme suivant :

1. $k = 0$
2. Extraction d'items fréquents d'Apriori sur une première fenêtre alloué sur la base de données.
3. Génération de candidats à partir des items et des motifs fréquents trouvés jusqu'ici et $k = k + 1$.
4. Glisser la fenêtre d'une transaction en avant (supprimer la première transaction et ajouter la transaction suivante dans la fenêtre).
5. Vérifier la fréquence des candidats de taille $k + 1$.
6. Reprendre en boucle à partir de l'étape 2.

En comparaison des modèles de fenêtres précédents, les fenêtres glissantes proposent de ne considérer la suppression d'éléments (et pas seulement l'ajout). L'avantage réside dans le fait que si une méthode fonctionne pour ce modèle, elle devrait fonctionner pour les modèles précédents (étant donné que les modèles précédents ne font pas de suppression). D'un autre côté, ajouter la gestion de l'historique sur toute la durée du flux implique une gestion de l'espace mémoire qui impacte obligatoirement l'observation sur la fenêtre la plus récente.

Fenêtre optimisant le support

Dans [TCG, CDG07] les auteurs proposent d'extraire les items (resp. les itemsets) fréquents dans un flux de données avec une gestion des fenêtres de temps différente des modèles précédent. L'idée de ces methodes d'extraction est d'optimiser la fréquence en fonction de la taille de la fenêtre d'observation. Ils proposent ainsi des fenêtres flexibles qui seront adaptées en fonction du support de chaque item(set) extrait. Dans leur définition du problème, les auteurs utilisent un modèle qu'on pourrait rapprocher du modèle de fenêtre à drapeau inversé. En effet il s'agit de considérer les fenêtres qui se terminent sur la date la plus récente (sorte de drapeau de fin, par opposition au drapeau de début). Ensuite leur algorithme détermine pour chaque

item(set) la taille de fenêtre qui optimise la fréquence de cet item(set) sur la période correspondant au début de la fenêtre jusqu'à la date actuelle. Une taille de fenêtre minimale mwl est utilisée comme paramètre d'entrée, avec un support minimum qui permet de réduire la taille du résultat. Considérons le support de a sur le flux d'items suivant : $< a\ b\ a\ a\ a\ b >$ et $mwl = 3$. Alors la fenêtre sur laquelle la fréquence de a est la plus élevée est la fenêtre de taille 4, qui termine le flux (*i.e.* $< a\ a\ a\ b >$) sur laquelle la fréquence de a est 3/4.

Notons que ces deux méthodes sont définies pour résoudre des problèmes qui sont proches de ceux étudiés dans la section suivante sur la découverte de rafales.

4.3.2 Découverte de rafales dans les flux

Ces dernières années, la détection de rafales (*bursts*) dans les flux de données a connu un intérêt grandissant. Un événement est considéré comme caractéristique d'une rafale si il apparaît avec un support très élevé dans une fenêtre temporelle précise. Dans ce sens, les méthodes de découverte de rafales dans les flux de données peuvent être considérées comme faisant partie des méthodes qui doivent gérer une fenêtre temporelle sur un flux afin de découvrir et/ou gérer des connaissances sur ces flux. Il y a beaucoup de définitions dans la littérature pour la notion de rafale, mais l'idée générale est de trouver des items qui correspondent à une fenêtre avec un support significatif. Il est intéressant de noter qu'à l'heure actuelle (et à notre connaissance) il n'existe pas de méthode de détection de rafale sous forme d'itemset (à l'exception de [CFL05] où les événements sont faits de corrélations entre items multiples).

Dans [CYL+05, MVY05, ZS03], on trouve des méthodes pour extraire les rafales sous la forme d'items fréquents. Dans la suite de cette section, je donne un résumé des méthodes proposées dans ces articles.

Dans [ZS03] le flux est une séquence unique d'items (ou d'images en 2D) et les auteurs proposent d'exploiter une méthode à base d'ondelettes afin de trouver les rafales. Dans leur cas, les items peuvent être des photons par exemple. Les auteurs introduisent la notion de fenêtre "élastique" et proposent une structure de données "Shifted Wavelet Tree" pour l'extraction de rafales. Leur algorithme est appliqué, entre autres, aux données issues de flux de données de rayons gamma (Milagro) ainsi qu'au New York Stock Exchange. Notons que la méthode proposée par [ZS03] permet de trouver la

taille exacte de la fenêtre temporelle sur laquelle la rafale se produit.

Les auteurs de [CYL+05] proposent également de manipuler un flux constitué d'une série d'items. Leur but est d'extraire des faux négatifs avec un algorithme (Loss-Negative) qui gère les rafales. Les faux-négatifs sont les items qui sont considérés comme non-fréquents alors qu'ils le sont. Les méthodes d'extraction d'items fréquents peuvent en effet reposer soit sur la gestion des faux-positifs, soit sur les faux-négatifs. Dans le premier cas, les méthodes gèrent la mémoire avec un paramètre d'erreur ϵ et considère chaque item dont le support est inférieur au support minimum s et supérieur à $s-\epsilon$ comme fréquent. Dans le second cas, les méthodes contrôlent la possibilité d'ignorer un item fréquent qui respecte le support minimum en utilisant un paramètre de fiabilité γ. L'avantage des méthodes basées sur les faux négatif réside dans le fait qu'elles reduisent l'espace combinatoire en ne maintenant pas tous les items qui ont un support entre s et $s-\epsilon$.

Les rafales peuvent aussi apparaitre dans des données financières. Ce type de rafales est étudié dans [MVY05] et les auteurs proposent de trouver des événements (items) qui apparaissent sous forme de rafales dans un flux fait de plusieurs séries temporelles. Les auteurs proposent alors l'identification de rafales sous la forme d'intervalles et la découvertes de rafales à partir d'une requête.

Dans [CFL05] on peut trouver la définition d'un problème de détection de rafales dans un flux de documents. Dans ce cas, le flux est fait de textes. Toutefois, les auteurs proposent d'utiliser l'information temporelle pour découvrir un ensemble de caractéristiques qui peuvent apparaître dans des fenêtres temporelles et de détecter les rafales sur la base de la distribution des caractéristiques avec l'algorithme HB-Event. Les événements sous forme de rafale sont faits de corrélations entre les caractéristiques (ce qui s'approche de la découverte d'itemsets). Cependant, HB-Event ne permet pas de découvrir des ensembles de caractéristiques (des itemsets) non disjoints. En d'autres termes lorsqu'une caractéristique (un item) est affecté à un ensemble, elle ne peut pas être affectée à un autre ensemble (impossible de trouver, par exemple, l'ensemble d'itemsets $E_1 = \{a, b\},\ E_2 = \{b, c\}$). Notons enfin que cette méthode travaille sur des fenêtres de taille fixe.

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. 4.5 – Etapes de l'alignement de séquences

4.4 Résumer un ensemble de séquences grâce à l'alignement

Dans nos travaux, nous avons utilisé l'alignement de séquences afin de résumer un flux de séquences de données. L'alignement de séquences d'itemsets a été étudié par [KPWD03, HWV02] pour la fouille de bases de données statiques. Il s'agit de constituer, à partir de plusieurs séquences, 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 représentée 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 6 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 6 *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 à l'alignement de ces séquences sont détaillées dans la figure 4.5. 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 4.5) est un résumé du cluster correspondant. Le motif séquentiel correspondant peut être obtenu en spécifiant k : le nombre minimum d'occurrences d'un item pour que celui-ci soit considéré comme fréquent. Par exemple, avec la séquence SA_{14} de la figure 4.5 et $k = 2$ la séquence alignée filtrée sera : <(a,b)(e)(h,i)(m,n)> (ce qui correspond aux items qui ont un nombre d'occurrences supérieur ou égal à k).

4.5 Résumer un flux de séquences et extraction de motifs séquentiels approximatifs

Cette section est basée sur l'article « Mining sequential patterns from data streams : a centroid approach » (Alice Marascu, Florent Masseglia) publié dans la revue internationale Journal of Intelligent Information Systems (JIIS) en 2006.

Dans [MM06] nous avons proposé SMDS. Le flux y est traité sous forme de batches (paquet de données) de taille fixe. Chaque batch contenant les séquences observées entre deux intervalles de temps. Soient $B_1, B_2, ...B_n$, les batches et B_n, le batch courant. Notre objectif est d'extraire les motifs sequentiels représentatifs de chaque batch b de $[B_1..B_n]$ et de stocker les motifs extraits dans une structure d'arbre préfixé (inspirée de [MCP98a]). Dans les grandes lignes, SMDS fonctionne de la manière suivante : classification de l'ensemble des séquences de chaque batch de transactions suivi d'un alignement pour chaque cluster ainsi obtenu. Cela permet d'obtenir des clusters de comportements qui représentent les usages du site en temps réel. Pour chaque cluster dont la taille est supérieure à $minSize$ (spécifié par l'utilisateur) SMDS ne stocke donc que le résumé du cluster. Ce résumé est donnée par l'algorithme d'alignement de séquences appliqué sur chaque cluster. La motivation de cette approche, basée sur l'alignement, vient du fait que l'extraction de motifs séquentiels traite un problème fortement combinatoire. Il est donc possible qu'un processus d'extraction exhaustif soit bloquant pour le flux. La section suivante explique les raisons de ces bloquages potentiels.

4.5.1 Limites de l'extraction de motifs séquentiels

Considérons que les motifs sont extraits par une méthode exhaustive (comme celles conçues pour les données statiques). Une telle méthode présentera au moins un type d'opération bloquante. Considérons par exemple le cas de l'algorithme PSP [MCP98a]. Nous avons testé cet algorithme sur des bases de données ne contenant que deux séquences (s_1 et s_2). Les deux séquences sont égales et contiennent des répétitions d'itemsets de taille 1. Plus précisément, la première base de test contenait onze répétitions des itemsets (1)(2) (*i.e.* $s_1 =< (1)(2)(1)(2)...(1)(2) >$, longueur($s_1$)=22 et $s_2 = s_1$). Le nombre de candidats générés à chaque passe est reporté dans la figure 4.6. La figure 4.6 reporte aussi le nombre de candidats générés pour les bases contenant des séquences de longueur 24, 26 et 28. On peut observer que, pour la

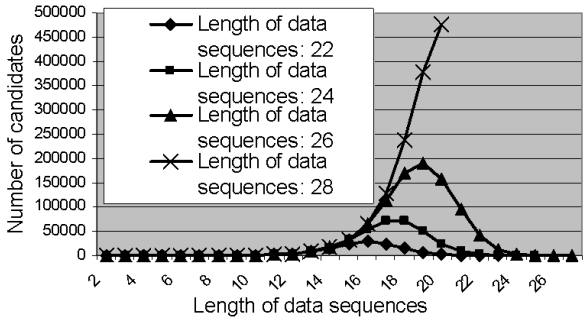

FIG. 4.6 – Limites d'un environnement intégrant PSP

base contenant des séquences de longueur 28, PSP est incapable de fournir les résultats (la mémoire est saturée par le nombre de candidats). Dans le contexte des flots de données issus des usages d'un site Web, il n'est pas rare de trouver de nombreuses répétitions d'un ou plusieurs items (fichiers pdf, php, etc.).

4.5.2 Algorithme glouton de classification des séquences

Notre premier schéma classificatoire est basique. Dans le but d'obtenir une classification des navigations aussi rapidement que possible, notre approche gloutonne 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. Aussitôt que n est similaire à une séquence de c alors n est insérée dans c. Si n n'est insérée dans aucun cluster, alors un nouveau cluster est crée et n est insérée dans ce nouveau cluster. Ce principe est illustré par la figure 4.7. La similitude entre deux séquences ($sim(s_1, s_2)$) est donnée dans la définition 17. s est insérée dans c si la condition suivante est respectée : $\exists s_c \in c / sim(s, s_c) \leq minSim$, avec $minSim$ la similitude minimum, spécifiée par l'utilisateur.

Définition 17 Soient s_1 et s_2 deux motifs séquentiels. Soit $|LCS(s_1, s_2)|$ la longueur de la plus longue sous-séquence commune entre s_1 et s_2. La similitude $sim(s_1, s_2)$ entre s_1 et s_2 est définie de la manière suivante : $sim = \frac{2 \times |LCS(s_1, s_2)|}{longueur(s_1) + longueur(s_2)}$.

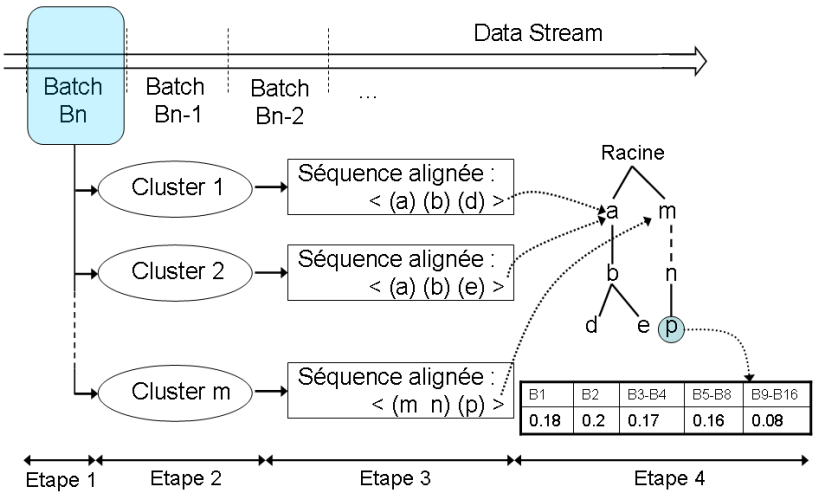

FIG. 4.7 – Vue d'ensemble des étapes de SMDS.

4.5.3 Complexité

Soit n le nombre de séquences du batch. Dans le pire des cas, l'algorithme de classification a une complexité en temps de $O(n^2)$. En fait, dans le pire des cas, LCS est appliqué une fois pour la première séquence, deux fois pour la deuxième, et ainsi de suite. La complexité est donc $O(\frac{n.(n+1)}{2}) = O(n^2)$. En ce qui concerne l'alignement, considérons que toutes les séquences d'un cluster C ont une longueur m et $|C| = p$. La complexité de l'alignement pour un batch est de $O(p.m^2)$.

Nos expérimentations ont montré que les temps d'exécution étaient suffisants pour répondre aux besoins d'analyse des flux de données. Cependant, la complexité du clustering reste élevée et peut être améliorée. Pour cela, la piste que nous avons suivie consiste à maintenir un centroïde pour chaque cluster. Ainsi, chaque nouvelle séquence peut être comparée aux centroïdes des clusters et non plus à toutes les séquences de tous les clusters existants. Malheureusement, cette idée se heurte à une difficulté importante : l'alignement ne peut pas être calculé de manière incrémentale. En effet, avant l'alignement, les séquences sont ordonnées afin d'optimiser le résultat. Comme les séquences arrivent dans les clusters par ordre de lecture, il n'est pas possible de les ordonner *a-priori*. Cela pose donc problème quand il s'agit de maintenir le centroïde des clusters. Le travail présenté dans la suite de cette

section a pour but de répondre à ce problème.

4.5.4 Approche centroïde pour la classification des séquences

Dans le but d'accélerer la classification des navigations, notre deuxième 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 le centroïde de 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 crée 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 4.5.4. 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 4.5.4. Enfin, l'ajout d'une séquence dans un cluster implique de mettre à jour son centroïde.

Calcul du centroïde

Le centroïde du cluster est déterminé par une technique d'alignement appliquée sur le cluster (comme [KPWD03, HWV02] 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 est décrit dans la section 4.4. À la fin du processus d'alignement, la séquence alignée (SA_{14} dans la figure 4.5) est considérée comme le centroïde du cluster. Dans SCDS, l'alignement se fait de manière incrémentale à chaque ajout d'une séquence dans le cluster. Pour cela nous maintenons une matrice de comptage des items dans chaque séquence et un tableau des distances entre chaque séquence et les autres. Ces éléments sont illustrés par la figure 4.8. La matrice (à gauche) stocke pour chaque séquence le nombre d'apparitions de chaque item dans cette séquence. Par exemple la séquence s_1 contient deux fois l'item a. Le tableau des distances stocke la somme des similitudes ($similMatrice$) entre chaque séquence et les autres séquences du cluster. Soit s_{1_i} le nombre d'apparitions de l'item i dans la séquence s_1 et m le nombre total d'items. $similMatrice$ est calculé grâce à la matrice de la manière suivante :

$similMatrice(s_1, s_2) = \sum_{i=1}^{m} min(s_{1_i}, s_{2_i})$.

Séq	a	b	c
s_1	2	0	1
s_2	1	0	1
⋮			

Séq	$\sum_{i=1}^{n} similMatrice(s, s_i)$
s_1	16
s_2	14
s_n	13
s_3	11
⋮	
s_{n-1}	1

FIG. 4.8 – Distances entre les séquences

Par exemple, avec les séquences s_1 et s_2 de la matrice donnée à la figure 4.8 cette somme vaut $s_{1_a} + s_{2_b} + s_{2_c} = 1 + 0 + 1 = 2$.

Cet alignement n'est cependant pas toujours calculé de manière incrémentale. Considérons l'ajout d'une séquence s_n. Tout d'abord s_n est ajoutée à la matrice et sa distance aux autres séquences est calculée ($\sum_{i=1}^{n} similMatrice(s_n, s_i)$). s_n est alors insérée dans le tableau de distances, en gardant l'ordre décroissant des valeurs de distances. Par exemple, dans la figure 4.8, s_n est insérée après s_2. Soit r le rang auquel s_n est insérée (dans notre exemple, $r = 2$) dans c. Il y a alors deux possibilités après l'insertion de s_n :

1. $r > 0.5 \times |c|$. Dans ce cas, l'alignement est calculé de manière incrémentale et
   $\varsigma_c = alignement(\varsigma_c, s_n)$.
2. $r \leq 0.5 \times |c|$. Dans ce cas il faut rafraîchir le centroïde du cluster et l'alignement est recalculé pour toutes les séquences du cluster.

Comparaison séquence/centroïde

Soit s la séquence à affecter dans un cluster et C l'ensemble des clusters. SCDS parcours 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 affecté à aucun cluster. Dans ce dernier cas, un nouveau cluster est crée et s y est affectée.

4.5.5 Expérimentations

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

Temps de réponse et robustesse de SCDS

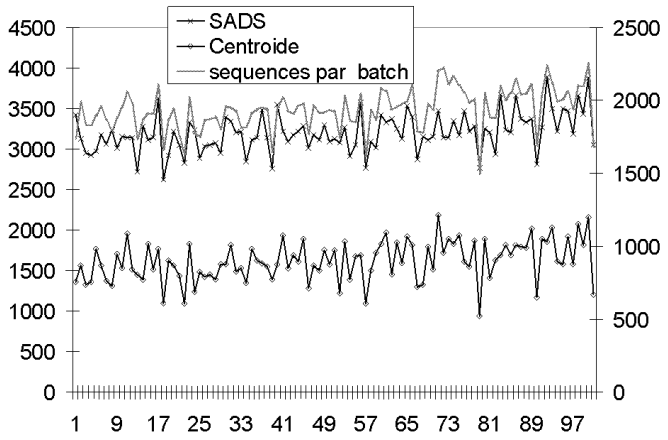

FIG. 4.9 – Temps d'exécution de SCDS.

Dans le but de montrer l'efficacité de SCDS, nous reportons à la figure 4.9 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 avons comparé ce temps de réponse à celui de SMDS [MM06] qui n'optimise pas la phase de clustering. En effet dans SMDS, la séquence à classer s est comparée à toutes les séquences de tous les clusters, jusqu'à ce que l'un des clusters présente une séquence compatible avec s. Nous pouvons observer que le temps de réponse de SCDS varie de 1000 ms à 2000 ms alors que le temps d'exécution de SMDS varie de 2500 ms à 4000 ms. Nous avons ajouté à la figure 4.9 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 et que SCDS demande 1400 ms pour en extraire les motifs séquentiels.

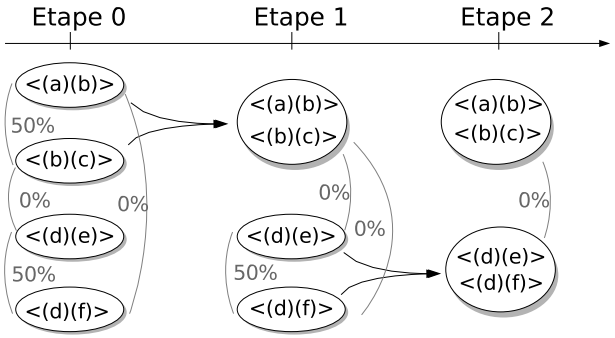

FIG. 4.10 – Clustering hiérarchique des séquences d'un batch.

Nous avons également implémenté un clustering hiérarchique sur les séquences de chaque batch. Le principe de ce clustering est décrit par la figure 4.10. Chaque séquence du batch est d'abord considérée comme un cluster (voir "étape 0" de la figure 4.10). A chaque étape la matrice des similitudes entre chaque cluster est calculée. Par exemple entre $<(a)(b)>$ et $<(b)(c)>$ la similitude est de 50%, les deux séquences partagent en effet la moitié de leurs informations. Entre $<(a)(b)>$ et $<(d)(e)>$ en revanche, la similitude est de 0%. Les deux séquences n'ont rien en commun. Les deux clusters les plus proches (ici il s'agit d'un ex-aequo entre $\{<(a)(b)>, <(b)(c)>\}$ d'un côté et $\{<(d)(e)>, <(d)(f)>\}$ de l'autre) sont regroupés en un seul cluster. L'étape "2" de la figure 4.10 nous montre en effet trois clusters : $\{<(a)(b)>, <(b)(c)>\}$, $\{<(d)(e)>\}$ et $\{<(d)(f)>\}$. Ce processus est alors réitéré jusqu'à ce que plus aucun cluster n'affiche une similitude supérieure à zéro avec au moins un des clusters restants. La dernière étape de la figure 4.10 nous

montre donc le résultat de cette classification : $\{<(a)(b)>, <(b)(c)>\}$ et $\{<(d)(e)>, <(d)(f)>\}$.

La comparaison avec les temps d'exécution du clustering hiérarchique est reportée à la figure 4.11 pour les 10 premiers batches. On peut y observer que SCDS obtient les résultats en un temps compris entre 1000 ms et 2000 ms. Le clustering hiérarchique nécessite entre 11000 ms et 17000 ms. Plus précisément, le temps d'exécution moyen de SCDS est de 1485 ms, contre 12451 ms pour le clustering hiérarchique.

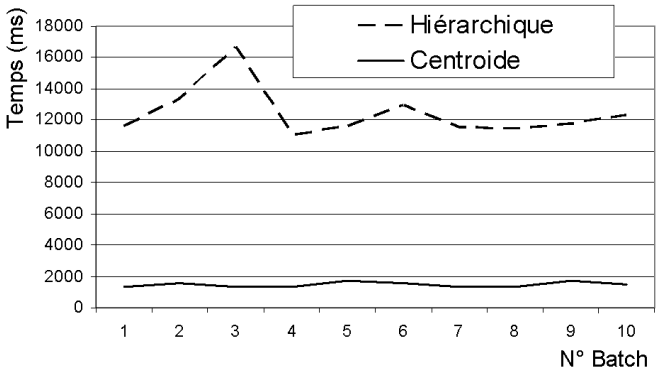

FIG. 4.11 – Temps de réponses du clustering hiérarchique et de SCDS pour 10 batches

Analyse de la qualité des clusters

Afin de mesurer la qualité des classes produites par SCDS, 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 17 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 4.12 la double moyenne DBM après avoir traité chaque batch. Soit C l'ensemble des classes, DBM est calculée de la manière suivante :

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

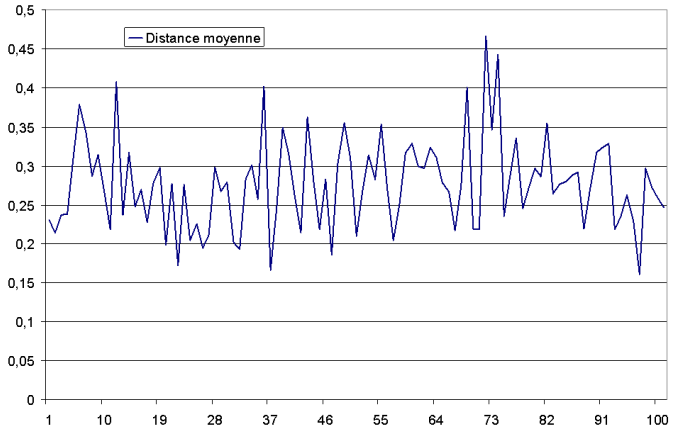

FIG. 4.12 – Distance globale, batch par batch

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 4.12. On peut y observer que DBM est comprise entre 15% et 45%. A la fin du processus, la valeur moyenne de DBM est de 28% (une qualité moyenne des classes de 72%).

Afin de compléter notre étude de la qualité des clusters obtenus avec SCDS, nous avons utilisé les mesures d'entropie et pureté, par comparaison avec les clusters obtenus par le clustering 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 du clustering 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 un bon clustering (par rapport au clustering de référence).

La pureté d'un cluster est donnée définie par : $P(C_r) = \frac{1}{n_r} max(n_r^i)$ et la pureté du clustering est donnée par la formule suivante : $Purete = \sum_{r=1}^{k} \frac{n_r}{n} P(C_r)$. Une grande valeur de pureté traduit un bon clustering par rapport au clustering 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	Pureté
1	0,012	0,951
2	0,013	0,943
3	0,017	0,932
4	0,015	0,936
5	0,016	0,932
6	0,018	0,927
7	0,02	0,928
8	0,015	0,938
9	0,017	0,933
10	0,018	0,93

On peut observer que la valeur de l'entropie se situe entre 0.012 et 0.02 et que la valeur de la pureté se situe entre 0.92 et 0.95. Lors de ces expérimentations, la valeur moyenne de l'entropie a été de 0.0166 et la valeur moyenne de la pureté a été de 0.9353.

4.6 Gestion de l'historique des connaissances

Une fois les motifs extraits, pour chaque batch dans le flux, il est nécessaire d'en gérer l'historique. Dans cette section, nous proposons un modèle basé sur l'optimisation de la mémoire disponible en fonction des besoins de chaque historique (par opposition aux modèles basés sur le vieillissement des données pour libérer la mémoire).

Considérons $T[1..n]$, l'ensemble des séries donné en section 4.2.2. À chaque étape s (chaque série est mise à jour avec une nouvelle valeur) nous voulons mettre à jour les représentations et les erreurs associées. Afin de maintenir un taux d'erreur global satisfaisant, nous devons choisir les représentations qui minimisent leur erreur locale. L'idée motivant ce choix est basée sur le fait qu'une représentation avec une erreur locale faible va mieux tolérer la fusion de deux segments (par rapport à une représentation avec une erreur déjà plus élevée). Quand $(s \times n) > M$ il faut libérer des blocs en mémoire afin de prendre en compte les nouvelles valeurs. Ainsi, à chaque étape s ; pour chaque nouvelle valeur :

1. Soit r la représentation ayant la plus faible erreur locale. Alors r est compressée (par la fusion de deux segments).
2. $E(r)$ l'erreur locale de r est mise à jour.
3. $E(R)$, l'erreur globale, est mise à jour.

Exemple 7 *La figure 4.13 donne une illustration de ce principe avec $n = 2$ (S_1 et S_2) et $M = 8$. Soit R_i la représentation de S_i. Deux nouvelles valeurs (à l'étape s de la figure 4.13) sont produites par le flux. R_1 et R_2 disposent chacune de 4 blocs (un segment étant représenté sur un bloc). Les trois étapes suivantes sont alors décrites par la figure 4.13 :*

1. *Suite à la production de ces deux valeurs il faut libérer deux segments. Étendre le segment $M4$ avec la nouvelle valeur de $S1$ n'a aucun coût en terme d'erreur. $M4$ est donc étendu avec la nouvelle valeur.*
2. *Le coût d'extension de $M8$ avec la nouvelle valeur de S_2 n'est pas négligeable. L'erreur résiduelle serait plus grande que celle résultant de la fusion de $M2$ et $M3$. Les deux erreurs sont décrites dans la figure 4.13. Ainsi, il est préférable (en terme d'erreur) de fusionner $M2$ et $M3$ et de réallouer le segment $M3$ à la représentation R_2.*
3. *$M2$ et $M3$ sont fusionnés en $M2$. $err1$ est mis à jour en $err1'$. $M3$ (maintenant libre) est alloué à R_2 et il est utilisé pour stocker la nouvelle valeur de S_2.*

H

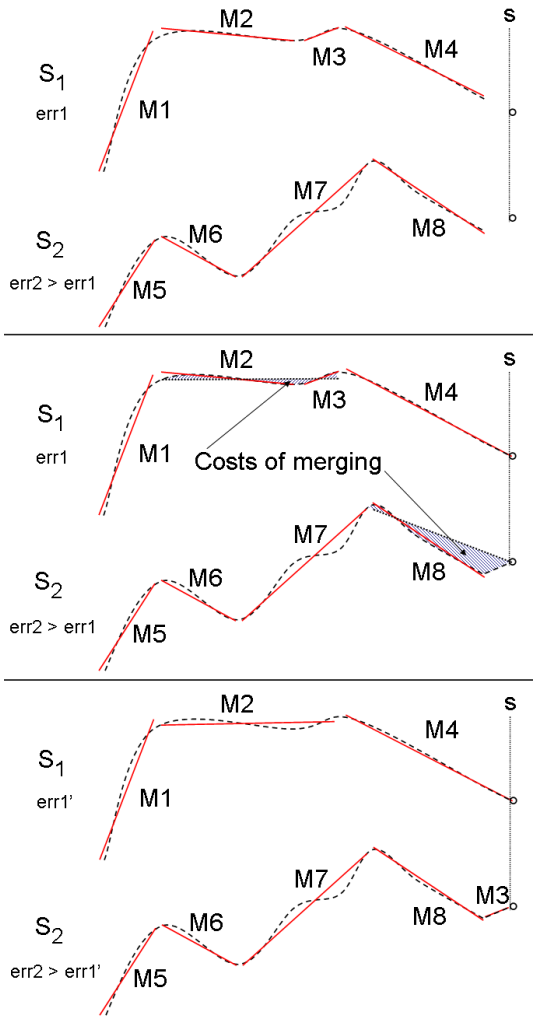

FIG. 4.13 – Principe général. Ici, les représentations partagent 8 blocs en mémoire qui sont alloués en fonction de l'erreur sur chaque représentation. La représentation de S1 n'a besoin que de 3 blocs alors que S2 en nécessite 5.

Après ces étapes, le processus peut continuer à mettre à jour le modèle en fonction des nouvelles valeurs du flux.

4.6.1 Motivation et verrous technologiques

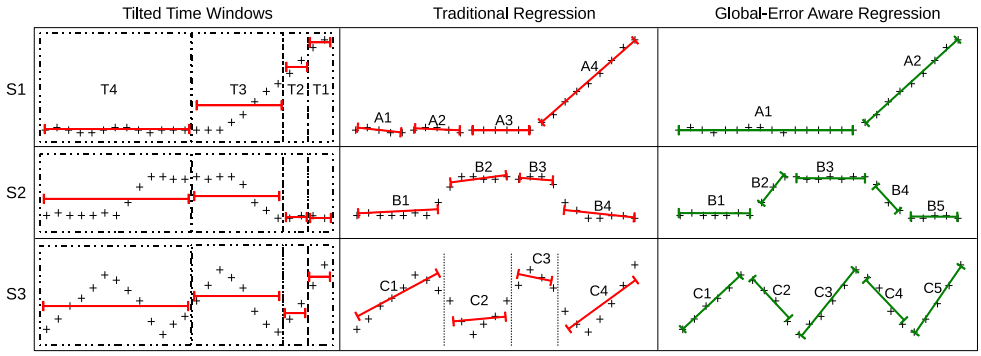

FIG. 4.14 – Un ensemble de séries temporelles et leur compression selon différentes techniques dans un espace mémoire de 12 blocs. L'approche optimisant l'erreur globale propose des approximations plus fidèles.

Les représentations basées sur un facteur d'ancienneté ont des avantages qui dépendent du contexte applicatif. Un de ces avantages est de donner plus d'importance aux événements récents. Toutefois, ce principe peut avoir des limites comme par exemple l'indifférence face aux changements importants dans les valeurs du flux. En effet, le facteur d'ancienneté est le seul critère utilisé lors de la compression rendant ce principe inadéquat pour certains types d'applications. La détection de fraudes, par exemple, peut nécessiter de trouver des anomalies dans l'historique du flux. Une anomalie est un événement souvent atypique (indépendamment de son ancienneté). Nous pensons que les requêtes sur l'historique d'un flux (stocké sous forme de représentation condensée) doivent se faire sur un modèle qui distribue la précision de la représentation en fonction de critères autres que l'ancienneté. Le besoin de précision (basé sur l'erreur résiduelle) est le critère privilégié dans ce travail.

La figure 4.14 illustre les avantages d'une régression basée sur l'erreur globale par rapport à un modèle utilisant un nombre égal de segments pour chaque représentation. Chaque modèle dispose de 12 segments en mémoire à distribuer entre les représentations. La RL ou les fenêtres logarithmiques n'essayent pas d'équilibrer cette distribution et allouent 4 segments à chacune

des trois représentations. La régression optimisant l'erreur globale distribue les segments de manière adéquate. Dans cette illustration, la représentation donnée par REGLO pour S_1 dispose de 2 segments alors que S_2 et S_3 disposent de 5 segments chacune. Ainsi, avec le même nombre de segments pour chaque modèle (RL, fenêtre logarithmiques et REGLO), l'erreur globale est significativement différente. Nous considérons que le tri des couples de segments par ordre de coût de fusion peut être géré par une pile. Ce principe repose sur une mesure fondamentale et complexe : calculer l'erreur résiduelle de la fusion de deux segments. *Dans un environnement dynamique tel qu'un flux de données, toute optimisation de cette évaluation est nécessaire.* La première optimisation est de rendre récursif le calcul de l'erreur résiduelle c'est-à-dire que l'erreur résiduelle de la fusion de deux segments se calcule en fonction de l'erreur résiduelle de chaque segment et du coût de fusion de ces deux segments. Nous comparons deux techniques possibles pour évaluer le coût de fusion d'un couple de segments : (1) La régression linéaire et (2) notre nouvelle technique d'approximation, basée sur les milieux des segments à fusionner. La section 4.6.2 donne les formules simplifiées du calcul de la RL sur deux segments en O(1) et du calcul de la nouvelle erreur en O(1). La section 4.6.3 donne les détails de notre approche pour la seconde technique d'approximation (basée sur les milieux) et explique comment évaluer l'erreur résiduelle avec cette technique. Chacune des méthodes présentées en sections 4.6.2 et 4.6.3 peut donc être utilisée pour calculer la fusion de deux segments et l'erreur résiduelle.

4.6.2 Nouvelles équations pour la RL

[CDH$^+$na] donne un ensemble de formules permettant d'obtenir la RL sur deux segments en O(1) et [PVKG08] donne la formule permettant de mettre à jour en O(1) l'erreur suite à cette fusion. Dans cette section, nous donnons des versions simplifiées de ces formules avec un nombre de variables et d'opérateurs significativement réduit. Nos formules permettront également une comparaison plus facile avec notre approche (présentée en section 4.6.3).

Tout d'abord, nous proposons une écriture simplifiée des formules permettant de fusionner deux segments en fonction de la pente et de la moyenne des valeurs des séries. Nous considérons le cas du modèle de régression linéaire $Y = \alpha.X + \beta$, où α est la pente ou le coefficient directeur de la droite régression et β est l'ordonnée à l'origine. Afin d'estimer α et β, nous avons $((i, j_i), i = 1, \ldots, s)$ une série temporelle sur un ensemble $S = \{i | i = 1, \ldots, s\}$. Soit $\bar{j}$ la moyenne des valeurs sur $(j_1, \ldots, j_s)$, alors $\bar{j} = \sum_{i=1}^{s} j_i/n$. L'idée de l'estimation des moindres carrés est de minimiser la somme des

erreurs carrées :$E(S) = \sum_{i=1}^{s} \varepsilon_i^2 = \sum_{i=1}^{s} (j_i - \alpha.i - \beta)^2$

Un ajustement linéaire pour une série temporelle j_i avec $i \in [1, s]$ est l'équation de régression linéaire $Y = \alpha.X + \beta$. Les paramètres α et β sont choisis pour minimiser $E(S)$ qui est la somme des carrés des erreurs résiduelles. Ces paramètres sont obtenus de la manière suivante :

$$\alpha = \frac{\sum_{i=1}^{s}(i - \bar{i})(j_i - \bar{j})}{\sum_{i=1}^{s}(i - \bar{i})^2} \; et \; \beta = \bar{j} - \alpha.\bar{i}$$

avec $\sum_{i=1}^{s}(i - \bar{i})^2 = (s-1)s(s+1)/12$ et $\bar{i} = (s+1)/2$.

La connaissance de α et de $\bar{j}$ permet de calculer les paramètres α et β de la RL. Afin d'accélérer les calculs, la mise à jour se fera uniquement sur les paramètres α et $\bar{j}$.

Fusion de deux segments

Considérons deux ensembles $S_1 = \{i | i = 1, \ldots, k\}$ et $S_2 = \{i | i = k+1, \ldots, s\}$ à fusionner. (S_1,S_2) est une bipartition de S. $Y = \alpha_1.X + \beta_1$ est l'équation de la régression linéaire de la série temporelle $((i, j_i), i = 1, \ldots, k)$.

Théorème 2 *Les paramètres α et $\bar{j}$ de S peuvent être dérivés des paramètres des ensembles S_1 et S_2 comme suit :*

$$\begin{aligned} S &= S_1 \cup S_2 \\ \alpha &= \frac{k^3 - k}{s^3 - s}.\alpha_1 + \frac{(s-k)^3 - (s-k)}{s^3 - s}.\alpha_2 - \frac{6k(s-k)}{s^3 - s}.(\bar{j}_1 - \bar{j}_2) \quad (4.1) \\ \bar{j} &= \frac{k.\bar{j}_1 + (s-k)\bar{j}_2}{s} \end{aligned}$$

Remarque : l'ordonnée β peut être calculée par la formule suivante :

$$\beta = \frac{k.\bar{j}_1 + (s-k)\bar{j}_2}{s} - \frac{s+1}{2}.\alpha$$

.

Preuve

Le numérateur et la pente α peuvent être écrit comme suit :

$$\sum_{i=1}^{k}(i - \bar{i}_1 + \bar{i}_1 - \bar{i})(j_i - \bar{j}_1 + \bar{j}_1 - \bar{j})$$
$$+ \sum_{i=k+1}^{s}(i - \bar{i}_2 + \bar{i}_2 - \bar{i})(j_i - \bar{j}_2 + \bar{j}_2 - \bar{j})$$

Le premier terme peut être décomposé en quatre termes :

$$\sum_{i=1}^{k}(i-\bar{i}_1)(j_i-\bar{j}_1)+k(\bar{i}_1-\bar{i})(\bar{j}_1-\bar{j})$$
$$+(\bar{j}_1-\bar{j})\sum_{i=1}^{k}(i-\bar{i}_1)+(\bar{i}_1-\bar{i})\sum_{i=1}^{k}(j_i-\bar{j}_1)$$
$$\text{on a}\sum_{i=1}^{k}(i-\bar{i}_1)=0 \text{ et } \sum_{i=1}^{k}(j_i-\bar{j}_1)=0$$

De plus, on a

$$\bar{i}_1-\bar{i} = \frac{s.\bar{i}_1-k.\bar{i}_1-(s-k)\bar{i}_2}{s}=\frac{(s-k)}{s}.(\bar{i}_1-\bar{i}_2)$$
$$\text{et} \quad \bar{j}_1-\bar{j} = \frac{(s-k)}{s}.(\bar{j}_1-\bar{j}_2)$$
$$\text{comme} \quad (\bar{i}_1-\bar{i}_2) = -s/2$$
$$\text{et} \quad (\bar{i}_1-\bar{i})(\bar{j}_1-\bar{j}) = \frac{-(s-k)^2}{2s}.(\bar{j}_1-\bar{j}_2)$$

Pour le numérateur et la pente on obtient :

$$(k^3-k).\alpha_1+((s-k)^3-(s-k)).\alpha_2$$
$$-\frac{k(s-k)^2}{2s}.(\bar{j}_1-\bar{j}_2)-\frac{k^2(s-k)}{2s}.(\bar{j}_1-\bar{j}_2)$$

Comme le dénominateur est égal à $(s-1)s(s+1)/12$ (ou $(s^3-s)/12$), le théorème est prouvé. Concernant le calcul de β, on part de la formule :

$$y_i=\alpha\cdot x_i+\beta, i\in[1,s]$$

On additionne tous les termes :

$$\sum_{i=1}^{s}y_i=\alpha\cdot\sum_{i=1}^{s}x_i+s.\beta, i\in[1,s]$$

et on divise l'équation par s.

Comme : $\frac{\sum_{i=1}^{s}y_i}{s}=\frac{\sum_{i=1}^{k}y_i+\sum_{i=k+1}^{s}y_i}{s}=\frac{k.\overline{j_1}+(s-k).\overline{j_2}}{k}$.

et $\frac{\sum_{i=1}^{s}x_i}{s}=\frac{\frac{s(s+1)}{2}}{s}=\frac{s+1}{2}$ On obtient : $\beta=\frac{k.\bar{j}_1+(s-k)\bar{j}_2}{s}-\frac{s+1}{2}.\alpha$

Ce théorème permet de construire une régression linéaire sur S avec les paramètres α_1, α_2, $\bar{j}_1$ et $\bar{j}_2$, à partir des régressions de S_1 et S_2 sans les points d'origine de la série j.

Mise à jour de l'erreur

Dans [PVKG08], le théorème 2 est important car il permet de calculer la somme des erreurs carrées sur S avec les erreurs carrées sur S_1 et S_2 et l'erreur entre les régressions locales sur S_1 et S_2 et la régression globale sur S. Ce qui s'exprime comme :

$$E(S) = E(S_1) + E(S_2) + \hat{E}(S_1 \cup S_2) \tag{4.2}$$

Selon le lemme 1 de [PVKG08], l'erreur $\hat{E}(S_1 \cup S_2)$ qui est le coût de cette fusion, peut être calculée en $O(1)$. Nous proposons une autre preuve de ce lemme avec une formule proche mais condensée qui dépend des paramètres des régressions linéaires de S_1 et S_2.

$$\begin{aligned} \hat{E}(S_1 \cup S_2) &= \sum_{i=1}^{k} (\alpha_1.i + \beta_1 - \alpha.i - \beta)^2 \\ &+ \sum_{i=k+1}^{s} (\alpha_2.i + \beta_2 - \alpha.i - \beta)^2 \end{aligned}$$

Ainsi, on obtient les formules suivantes :

$$\begin{aligned} \hat{E}(S_1 \cup S_2) &= k(\beta_1 - \beta)^2 + k(k+1)(\alpha_1 - \alpha)(\beta_1 - \beta) \\ &+ \frac{k(k+1)(2k+1)}{6}(\alpha_1 - \alpha)^2 \\ &+ (s-k)(\beta_2 - \beta)^2 \\ &+ (s-k)(s-k+1)(\alpha_2 - \alpha)(\beta_2 - \beta) \\ &+ \frac{(s-k)(s-k+1)(2(s-k)+1)}{6}(\alpha_2 - \alpha)^2 \end{aligned}$$

La mise à jour de l'erreur $E(S)$ permet de calculer le coût de la fusion de S_1 avec S_2. Ce coût est égal à $C(S_1 \cup S_2) = E(S) - E(S_1) - E(S_2) = \hat{E}(S_1 \cup S_2)$ et est utilisé dans l'étape 2 de l'algorithme REGLO.

4.6.3 Une approche rapide pour la fusion des segments

Bien que les résultats précédents permettent de fusionner et d'évaluer l'erreur et le coût de fusion en O(1), nous voulons optimiser encore cette étape. En effet, de meilleurs temps d'exécution peuvent être obtenus en diminuant le nombre d'opérations, ce qui peut être crucial dans le contexte des flux de données. Dans cette section, nous proposons de calculer une approximation de la RL grâce à une idée innovante basée sur l'observation suivante :

quand le coût de fusion de deux segments est faible alors les points de croisements entre la droite de régression et les segments sont proches des milieux de ces segments.

Fusion de deux segments

La RL a pour résultat une représentation d'une série temporelle telle que chaque segment minimise la somme des erreurs quadratiques entre la représentation et les données d'origine. Notre approche consiste à élaborer une approximation intuitive et efficace de ces valeurs d'origine. AMi, notre technique d'approximation, sera représentée par la ligne qui coupe les segments d'origine en leurs milieux. Cette technique présente deux avantages principaux :

1. Elle ne demande pas de nombreux calculs impliquant autant de variables que la RL comme exprimée dans le théorème 2, ce qui la rend plus rapide que cette dernière.
2. Son interprétation est naturelle. Cette technique ne minimise pas la somme du carré des erreurs, mais nous montrons que l'approximation obtenue est très fidèle aux données d'origine. En effet, la somme du carré des erreurs a pour but de pénaliser les représentations qui ne tiennent pas compte des valeurs aberrantes. L'erreur sur ces valeurs étant rendues très importante dans le calcul global en raison de la puissance de deux utilisée. Privilégier les valeurs aberrantes n'est pas forcément le meilleur moyen d'avoir une représentation fidèle des données d'origine. Comme nous le montrerons, les représentations obtenues par AMi et la RL sont comparables. Toutefois, nous voulons être moins sensibles aux valeurs aberrantes. Ainsi, la représentation obtenue par AMi est un bon compromis entre les normes $\mathcal{L}_1$ et $\mathcal{L}_2$. En effet, nos expérimentations montrent que AMi peut avoir de meilleurs résultats quand l'erreur est mesurée dans la norme $\mathcal{L}_1$.

Exemple 8 *La figure 4.15 illustre la différence entre la RL (gauche) et AMi (droite). Avec AMi, un premier ensemble de 4 segments est construit entre les points 1 et 2, les points 3 et 4, les points 5 et 6 et les points 7 et 8. Ensuite, le deuxième niveau d'approximation fusionne les deux premiers segments en passant par les milieux des segments et on obtient le segment* $S1$ *de la figure 4.15. Le même principe est appliqué aux deux segments restants (points 5 à 8) pour donner le segment* $S2$*. Le troisième niveau d'approximation fusionnera les deux segments* $S1$ *et* $S2$ *avec pour résultat* S*, le segment qui passe par leurs milieux (i.e.* $(\bar{i}_1, \bar{j}_1)$ *et* $(\bar{i}_2, \bar{j}_2)$*).*

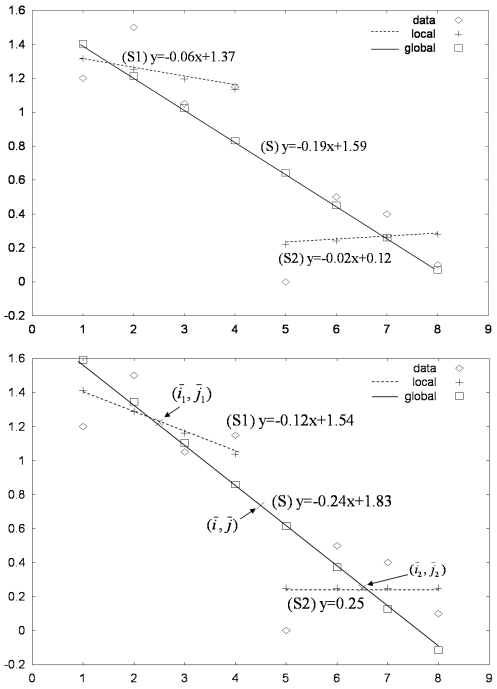

FIG. 4.15 – Illustration de la RL (minimisant l'erreur quadratique) en comparaison d'AMi (ligne qui passe par les milieux des segments).

Soit $(\bar{i}_1, \bar{j}_1)$ et $(\bar{i}_2, \bar{j}_2)$ les milieux des segments $S1$ et $S2$ avec $\bar{i}_1 = (k+1)/2$ et $\bar{i}_2 = (s+k+1)/2$. L'équation du segment S est obtenue par résolution de l'ensemble d'équations linéaires suivant :

$$\bar{j}_1 = \alpha.\bar{i}_1 + \beta \text{ et } \bar{j}_2 = \alpha.\bar{i}_2 + \beta$$

La pente α et l'ordonnée β sont données par :

$$\alpha = \frac{\bar{j}_1 - \bar{j}_2}{\bar{i}_1 - \bar{i}_2} \text{ et } \beta = \frac{\bar{j}_2.\bar{i}_1 - \bar{j}_1.\bar{i}_2}{\bar{i}_1 - \bar{i}_2}$$

et le nouvau point milieu $(\bar{i}, \bar{j})$ de S est donné par : $\bar{i} = \frac{s+1}{2}$ et $\bar{j} = \alpha.\bar{i} + \beta$
Le calcul de $\bar{j}$ est donné par : $\bar{j} = \frac{k.\bar{j}_1 + (s-k)\bar{j}_2}{s}$

Dans le cas où l'hypothèse de la distribution uniforme sur S est vérifiée, le point milieu de S est le point moyen et le point médian. Cette hypothèse est vérifiée car nos observations arrivent par intervalles réguliers. Le point

milieu de S vérifie les équations suivantes :

$$\bar{i} = \frac{1}{s}.\sum_{i=1}^{s} i \text{ et } \bar{j} = \frac{1}{s}.\sum_{i=1}^{s} j_i$$

De ce fait, ce point appartient à la droite de régression linéaire. Le théorème 3 montre la relation entre les pentes de la RL et de AMi, qui peut être calculé par :

$$\alpha^{RL} = \frac{k^3 - k}{s^3 - s}.\alpha_1^{RL} + \frac{(s-k)^3 - (s-k)}{s^3 - s}.\alpha_2^{RL} + \frac{3k(s-k)}{s^2 - 1}.\alpha^{AMi}$$

Mise à jour de l'erreur

Théorème 3 *L'erreur commise par AMi sur le segment $S = S_1 \cup S_2$ peut être calculée de manière récursive à partir des erreurs pondérées des segments S_1 et S_2 et de la pente α du segment S.*

Preuve

La somme des erreurs carrées de S peut être décomposée en deux parties :

$$E(S) = \sum_{i=1}^{k}(j_i - \alpha.i - \beta)^2 + \sum_{i=k+1}^{s}(j_i - \alpha.i - \beta)^2$$

La première partie de cette décomposition peut être réécrite de la manière suivante :

$$\begin{aligned}
&\sum_{i=1}^{k}(j_i - \alpha.i - \beta - (\alpha_1.i + \beta_1) + (\alpha_1.i + \beta_1))^2 \\
&= \sum_{i=1}^{k}(j_i - \alpha_1.i - \beta_1)^2 + (\alpha_1.i + \beta_1 - \alpha.i - \beta)^2 \\
&\quad +2(j_i - \alpha_1.i - \beta_1)\big((\alpha_1 - \alpha).i + (\beta_1 - \beta)\big)
\end{aligned}$$

La première somme des erreurs quadratiques dans la première ligne de cette réécriture représente la somme des erreurs quadratiques de S_1 obtenue avec AMi. La seconde partie représente les erreurs entre le segment S_1 obtenu par AMi et le segment S obtenu par AMi calculé sur S_1. Comme le modèle obtenu par la RL, AMi vérifie que $\sum_{i=1}^{k}(j_i - \alpha_1.i - \beta_1) = 0$ et la dernière

partie peut être simplifiée par :

$$\begin{aligned}\Delta(S_1/\alpha) &= 2(\alpha_1-\alpha)\sum_{i=1}^{k}(j_i-\alpha_1.i-\beta_1)i \\ &= 2(\alpha_1-\alpha)\sum_{i=1}^{k} i.j_i - k(\alpha_1-\alpha)\big(\alpha_1(2k+1)(k+1)/3-\beta_1(k+1)\big)\end{aligned}$$

Avec AMi, la somme des erreurs quadratiques de $E(S)$ est plus complexe que la somme des erreurs quadratiques de la RL définie par l'équation 4.2 mais la complexité par rapport à s est inchangée. $E(S)$ peut être exprimé de la manière suivante :

$$E(S) = E(S_1)+E(S_2)+\hat{E}(S_1\cup S_2)+\Delta(S_1/\alpha)+\Delta(S_2/\alpha) \quad (4.3)$$

Le calcul de $\Delta(S_1/\alpha)$ ou $\Delta(S_2/\alpha)$ dépend de la pente du nouveau segment S et des valeurs $\delta(S_1)=\sum_{i=1}^{k} i.j_i$ et $\delta(S_2)=\sum_{i=k+1}^{s} i.j_i$. Ainsi, la relation linéaire $\delta(S)=\delta(S_1)+\delta(S_2)$ existe et δ est maintenu pour chaque segment. De plus, $\Delta(S_1/\alpha)$ peut se calculer facilement :

$$\Delta(S_1/\alpha) = -\alpha.\delta(S_1)+\Lambda(S_1)$$

Où $\Lambda(S_1)$ dépend seulement du segment S_1 et on obtient :

$$\begin{aligned}E(S) &= (E(S_1)+\Lambda(S_1))+(E(S_2)+\Lambda(S_2)) \\ &+ \hat{E}(S_1\cup S_2)-\alpha(\delta(S_1)+\delta(S_2))\end{aligned}$$

4.6.4 Expérimentations

Dans cette section, nous évaluons l'efficacité et la qualité de notre méthode d'approximation grâce à un ensemble d'expérimentations. Nous avons implémenté deux versions de REGLO. Dans la première version, l'approximation et le calcul de l'erreur sont basés sur la RL (telles que présentées en section 4.6.2). Dans la deuxième version, l'approximation et le calcul de l'erreur sont basés sur AMi (C.f. section 4.6.3). Nous avons également implémenté une compression basée sur les ondelettes et une autre basée sur les fenêtres logarithmiques afin de comparer REGLO avec les techniques les plus importantes. Une description des ondelettes peut être trouvée en [Dau92]. Les fenêtres logarithmiques sont décrites en [CDH$^+$na, GHP$^+$03]. Nous avons évalué nos algorithmes sur deux jeux de données réelles. Le premier jeu de données, "NYSE", est construit sur les données téléchargées

depuis `http://icf.som.yale.edu/nyse`. Ce jeu contient 134 séries temporelles, chacune contenant 804 valeurs correspondant aux valeurs du marché de New-York de 1815 à 1925. Le deuxième jeu de données, "WEB", vient des logs d'accès Web anonymisés de l'Inria Sophia-Antipolis. Nous avons recueilli un an d'usage, pour un total de 14 Go. Un premier prétraitement a permis d'extraire les URLs les plus fréquentes de ce jeu. Avec un support minimum de 1%, nous en avons trouvé 223. Ensuite, nous avons reporté les variations du nombre de requêtes à ces URLs avec une fenêtre sautante de taille 1000. En d'autres termes, chaque série est mise à jour toutes les 1000 requêtes et la valeur affectée à la série correspond au nombre de requêtes sur l'URL pendant cet intervalle.

REGLO (versions RL et AMi) est implémenté en Java. Les expérimentations sont effectuées sur un PC équipé de 2 Gb de mémoire et un pentium à 4 processeurs cadencés à 2,2 Ghz. *Nos algorithmes et jeux de données sont disponibles et nos expérimentations peuvent être répétées.*

Comparaison entre AMi et la RL

Dans ce premier ensemble d'expérimentations nous évaluons, sur les deux jeux de données :

- L'impact de AMi sur l'erreur moyenne à chaque étape.
- La différence des temps d'exécutions entre AMi et la RL.

La figure 4.16 montre l'erreur moyenne des approximations obtenues par AMi et la RL pour les données NYSE (haut) et WEB (bas) sur les 251 premières valeurs. L'erreur est calculée de la manière suivante : $\forall i \in [1..s] err[i] = averageError_{j=1}^{j=n}(R[j][i])$ où $R[j][i]$ est l'erreur entre la représentation de l'enregistrement i de la série j et la valeur réelle de i dans les données d'origine. Ainsi, pour chaque unité de temps, nous connaissons l'erreur moyenne du modèle pour les n séries temporelles. Pour NYSE les erreurs moyennes commises par AMi et la RL sont très proches. La qualité de l'approximation obtenue par AMi est illustrée sur le jeu de données WEB. Pour ce jeu, AMi obtient une très bonne erreur moyenne en comparaison de la RL. La sous-section 4.6.4 donne les erreurs globales moyennes des deux méthodes d'approximation et AMi obtient de meilleurs résultats (0,87) que la RL (2,52). Cela est dû au fait que AMi ne tente pas d'optimiser la somme des erreurs carrées. En effet, notre but est trouver un compromis entre l'erreur minimale en $\mathcal{L}_1$ et celle en $\mathcal{L}_2$, ce pour quoi AMi obtient des résultats meilleurs que la RL (comme illustré par la figure 4.16).

La figure 4.17 montre les temps d'exécution de REGLO sur les deux jeux de données, selon que l'on utilise l'approximation de AMi ou de la RL. Les

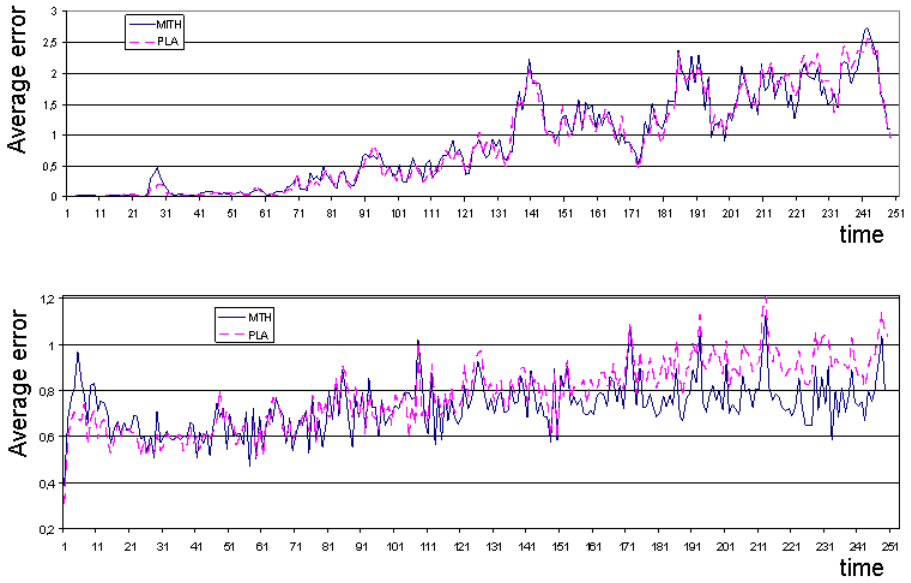

FIG. 4.16 – Erreur moyenne étape par étape pour AMi et la RL sur NYSE et WEB.

temps de réponse sont systématiquement plus faibles pour AMi. L'explication vient du nombre considérablement réduit d'opérations (Cf. les formules de la section 4.6.3) nécessaires à AMi pour calculer l'approximation et l'erreur résiduelle en comparaison des formules complexes (que nous avons pourtant simplifiées autant que nous le pouvons) de la RL (Cf. section 4.6.2).

TAB. 4.1 – Erreurs globales des ondelettes, des fenêtres logarithmiques, de la RL et de AMi

	WEB	NYSE
Fenêtres	17.245627	2.451278
Ondelettes	14.0588	1.55812
RL	2.529895911	0.857884432
AMi	0.879335223	0.868300858

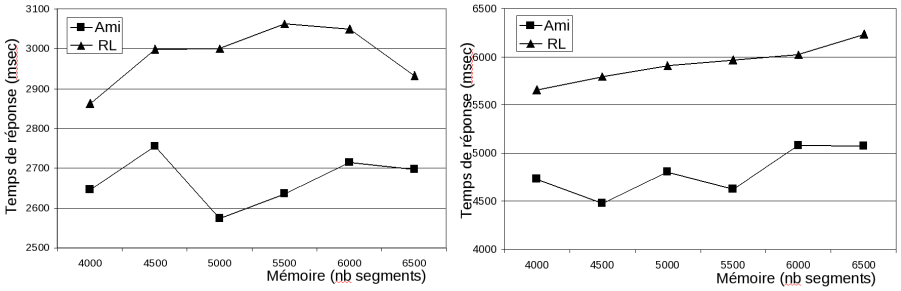

FIG. 4.17 – Temps d'exécution de AMi et de la RL sur NYSE et WEB en fonction de la mémoire disponible.

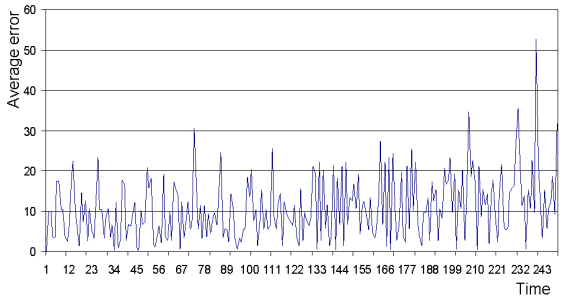

FIG. 4.18 – Erreur moyenne, étape par étape, des ondelettes sur WEB.

Ondelettes et fenêtres logarithmiques

L'objectif de cette section est de comparer l'erreur commise par REGLO avec celles des ondelettes et des fenêtres logarithmiques. La figure 4.18 montre l'erreur moyenne à chaque étape (principe similaire à la figure 4.16) avec une représentation basée sur les ondelettes de Haar sur les 250 premières valeurs. Dans le cas de la figure 4.18, chaque série dispose de 34 segments. Ce nombre correspond à la mémoire disponible à la figure 4.16. En d'autres termes, pour chaque représentation par ondelettes on conserve les 34 coefficients les plus significatifs.

Concluons cette section en observant l'erreur globale commise par chaque représentation. Le tableau 4.1 donne l'erreur globale moyenne de chaque modèle en comparaison aux données d'origine. Pour chaque modèle, l'erreur moyenne est calculée à chaque étape (comme décrit à la section 4.6.4). En-

suite, la valeur moyenne des erreurs de chaque modèle est reportée dans la table 4.1. Les modèles obtenus par AMi et la RL disposent chacun de 5000 segments. Les modèles basés sur les fenêtres logarithmiques demandent 11 segments par série (11 étant obtenu par le logarithme de s, le nombre maximum de valeurs dans les séries du jeu de données). Le modèle basé sur les ondelettes dispose d'un nombre de segments correspondant à celui de REGLO (*c.à.d.* M/n coefficients pour chaque série, avec M le nombre de segments disponibles dans REGLO et n le nombre de séries à modéliser).

4.7 Discussion

Ce chapitre a présenté deux approches destinés à :
- Extraire des connaissances dans les flux de données.
- Gérer l'historique de ces connaissances.

Je propose, dans cette section, des discussions sur ces deux aspects.

4.7.1 Conserver les centroïdes d'un batch à l'autre

Dans la section 4.5 nous avons proposé un principe conçu pour extraire rapidement les séquences d'un data stream et en proposer un résumé significatif. Notre premier algorithme repose sur une technique de classification combinée avec un alignement des séquences. Grâce à cette façon de traiter le flux, SMDS est capable de détecter des comportement partagé par un nombre relativement faible d'utilisateurs (*e.g.* 13 utilisateurs ou encore $0,1\%$) ce qui est proche du difficile problème de l'extraction de motifs séquentiels avec un support très faible. Dans un deuxième temps, nous avons proposé la méthode SCDS pour améliorer la complexité de SMDS. Dans ce cas, le clustering est basé 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. Nos expérimentations ont montré que SCDS traite le flux assez rapidement pour être intégré dans un contexte temps réel.

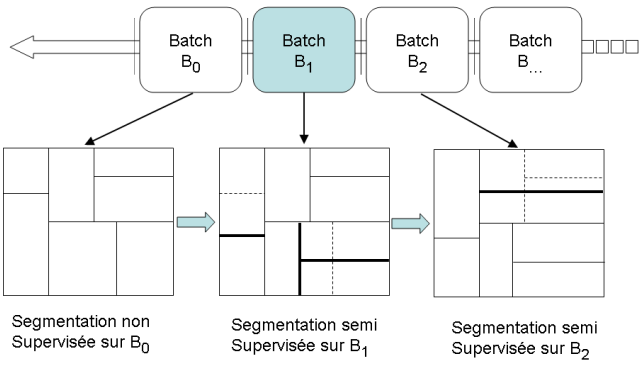

FIG. 4.19 – Classification incrémentale dans un flux de données

Dans [MM07] nous avons égakement proposé ICDS, un troisième algorithme, non détaillé dans ce mémoire. Cette partie de la discussion est consacrée à cette proposition. Le principe d'ICDS, illustré par la figure 4.19 est

de garder les centroïdes des clusters d'un batch à l'autre. L'idée étant que les centroïdes seront globalement similaires entre deux batches et que l'on peut garder une représentation globale (les centroïdes) du clustering au batch B_{n-1} au moment de traiter le batch B_n. À la fin du traitement de B_n, la classification peut avoir évolué par rapport à celle 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. L'idée de base était de gérer l'évolution, dans le flux, des contenus des clusters. Ainsi, il est possible d'obtenir un historique du support du cluster mais également un historique de son centroïde.

Concernant les temps de réponse, lors de nos expérimentations, nous avons constaté qu'ils étaient similaires à ceux de SCDS. L'explication vient du fait que dans SCDS les premiers clusters se construisent très rapidement (il y a peu de comparaisons à faire). D'un autre côté, dans ICDS, les clusters sont déjà tous présents et il faut tous les prendre en compte avant d'affecter une séquence à un cluster. Le temps de mise à jour des centroïdes économisé est donc compensé par un grand nombre de clusters auxquels comparer les nouvelles séquences.

4.7.2 Gestion de l'historique et normes de mesure de l'erreur

Pour gérer et résumer les historiques de fréquence des motifs extraits nous avons présenté (1) REGLO, une stratégie de résumé d'un flux de séries temporelles et (2) AMi, un principe rapide d'approximation des valeurs d'une série. REGLO est capable de gérer de nombreuses séries temporelles sans limite de taille grâce à son principe d'équilibre et de distribution de la mémoire entre les représentations. Ce principe permet à chaque série d'obtenir ou de restituer de l'espace mémoire selon ses besoins de précision. Notre principe d'approximation rapide est comparé à la régression linéaire dans ses aspects formels mais aussi grâce à une batterie d'expérimentations. Notre

étude sur les relations entre AMi et la RL montre que AMi permet un calcul récursif de l'erreur entre la représentation et les données d'origine.

Dans nos expérimentations, l'erreur moyenne de chaque représentation (AMI et la RL) est mesurée en $\mathcal{L}_1$. Il est évident qu'en $\mathcal{L}_2$ cette erreur serait meilleure pour la RL, car cette représentation est conçue pour obtenir la meilleure erreur possible dans cette norme. Toutefois, une erreur optimisée en $\mathcal{L}_2$ n'est pas forcément le meilleur choix dans notre cas. En effet, l'erreur quadratique est fortement influencée par les valeurs abherrantes. Or, dans le modèle REGLO, les valeurs abherrantes sont justement prises en compte pour éviter leur compression. Si des fluctuations rapides et importantes se présentent, elles seront privilégiées par REGLO en préférant comprimer des segments plats. Dans notre cas il serait donc préférable d'optimiser une erreur calculée en $\mathcal{L}_1$. Ce n'est pas le cas pour l'instant, dans la mesure ou AMI optimise une sorte de compromis entre les deux normes. Une piste de recherche consisterait à définir une approximation optimisant l'erreur en $\mathcal{L}_1$ avec un temps de calcul aussi rapide que celui de AMI.

4.7.3 Calculs distribués en temps réel

Dans [MPT06] nous avons proposé une méthode pour extraire des comportements fréquents dans un réseau peer-to-peer. L'objectif était de déterminer les comportements fréquents sur les nœuds du réseau afin d'optimiser les échanges. Par exemple, en examinant les requêtes sur les nœuds, on peut découvrir que 77% des nœuds qui font une requête pour *Mandriva Linux* vont télécharger le fichier *Mandriva Linux 2005 CD1 i585-Limited-Edition-Mini.iso* puis feront une nouvelle requête avec tous les noms possibles des images iso concernées par la distribution Linux recherchée, à savoir *Mandriva Linux 2005 Limited Edition*. Puis, parmis les nombreuses réponses à cette requête, le fichier *Mandriva Linux 2005 CD2 i585-Limited-Edition-Mini.iso* est téléchargé.

Pour réaliser cette extraction, nous avons alors proposé un schéma de calcul distribué sur les nœuds et coordonnés par un nœud central. Ces résultats sont obtenus en temps réel grâce à une heuristique de calcul et de d'estimation des motifs fréquents.

4.7.4 Les séquences ne sont pas atomiques

Les travaux présentés dans ce chapitre proposent de découper le flux en batches de séquences. Ce principe permet de tester des méthodes et de

réaliser des expérimentations mais manque encore de réalisme. En effet les séquences se constituent par accumulation dans le temps et le découpage par batches découpe les séquences pendant leur construction. Avec la thèse de Chongsheng Zhang (soutenance prévue en septembre 2011) nous travaillons sur un modèle préservant la construction par accumulation des objets d'un flux, tout en conservant la possibilité d'en extraire des connaissances.

Chapitre 5

Fouille de données, flux et sécurité : aux évolutions précédentes s'ajoute celle des comportements malicieux

La détection d'intrusions est un sujet important pour la sécurité des réseaux auquel sont consacrés de nombreux travaux [LS98, VS01, LEK+03, PP07]. Les systèmes de détection d'intrusion ou IDS (*Intrusion Detection Systems*) ont pour objectif la protection des systèmes d'information contre les intrusions et les attaques. Ils sont traditionnellement basés sur les signatures d'attaques connues [Roe98, BWJ01]. Ansi, les nouvelles attaques sont régulièrement ajoutées dans la base de signatures. Le principal défaut de ces approches est leur manque de réactivité face à de nouveaux types d'attaques. Une nouvelle attaque, par exemple basée sur la découverte récente d'une faille, sera probablement ignorée par l'IDS qui n'a encore aucune connaissance dans sa liste lui permettant de la détecter.

Protéger un système contre les nouvelles attaques, avec une réactivité satisfaisante, est un sujet important dans ce domaine et la fouille de données pourrait être une source de solutions. En effet, la fouille de données pour la détection d'intrusion est un sujet qui permet de proposer de nouveaux outils afin de détecter les comportements malveillants [Luoty, DEK+02, BCH+01, MCZH00, WZ03]. Parmi ces approches, les modèles prédictifs ont pour objectif d'améliorer la base de signatures d'un IDS [WZ03]. La fouille de données peut également proposer des solutions pour détecter les anomalies et en dé-

duire les intrusions [LEK+03, EAP+02, CAMSC05]. Dans ce cas, le principe est généralement d'obtenir une classification (des clusters) sur les données et d'en isoler les atypiques (ou *outliers*, ces événements qui ne correspondent à aucune classe identifiant un usage normal). En effet, la détection d'outliers permet d'isoler les enregistrements qui dévient de manière significative d'une notion de normalité bien définie. Ses champs d'application sont nombreux, allant de la détection de fraudes pour les cartes de crédit [AFR97] à la cybersécurité [BCH+01] en passant par la sécurité des systèmes critiques [FYM05].

Toutefois le principal inconvénient des systèmes de détection d'intrusion basés sur les outliers se trouve dans leur taux de faux positifs extrêmement élevé. En effet, dans ces systèmes, une alarme peut être déclenchée à cause d'une nouvelle classe de comportements normaux mais encore inconnus. Si on considère le grand nombre de nouveaux usages dans les systèmes actuels (par exemple, un site d'enchère en ligne, un site d'information ou encore un site de socialisation ont en commun l'émergence rapide et fréquente d'usages nouveaux) un taux de faux positifs même très faible donnera un nombre d'alarmes ingérable pour l'analyste.

Ce chapitre résume des communications publiées sur ce sujet dans lesquelles notre objectif est de :

1. Valider une approche de **détection des outliers communs à deux sites** et estimer leur degré de dangerosité (C.f section 5.4).
2. **Détecter les outliers dans un contexte de flux de données** sans faire intervenir l'utilisateur (C.f section 5.5).

5.1 Activités

Je suis porteur de deux projets sur le thème des flux de données et de la sécurité :

- ARC Inria « SéSur » (2006-2008) avec le LIRMM, le LGI2P et l'IRISA (budget global du projet : 24KEuros).
- Projet Color Inria « Mutan » (2008) avec le LGI2P (12KEuros).

5.1.1 Encadrement

- **Doctorants** :
  - Je suis directeur de thèse (par dérogation de l'Université de Nice

Sophia-Antipolis) de Chongsheng Zhang depuis septembre 2008. Financement ANR et Inria.
  - J'ai participé à l'encadrement d'Alice Marascu (sous la direction d'Yves Lechevallier, Inria). C.f. la section 4.1.1
- **Post-doc** :
  - Wei Wang, co-encadré avec Marie-Odile Cordier, René Quiniou (IRISA) et Brigitte Trousse. Financement Inria (campagne interne). Taux de participation : 25%.
  - J'ai participé à l'encadrement de Céline Fiot. C.f. la section 3.1.1.

5.1.2 Publications

Les travaux décrits dans ce chapitre ont donné lieu à des publications dans trois conférences internationales, dans un ouvrage international et dans une conférence nationale :

- Alice Marascu and Florent Masseglia. « A Multi-Resolution Approach for Atypical Behaviour Mining ». In 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'09). Bangkok, Thailand, April 2009.
- G. Singh, F. Masseglia, C. Fiot, A. Marascu and P. Poncelet. « Data Mining for Intrusion Detection : from Outliers to True Intrusions. » In 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'09). Bangkok, Thailand, April 2009.
- Alice Marascu and Florent Masseglia. « Parameterless Outlier Detection in Data Streams ». In 24th Annual ACM Symposium on Applied Computing (ACM SAC'09). Honolulu, Hawaii, USA, March 2009.
- Wang, W., Masseglia, F., Guyet, T., Quiniou, R., and Cordier, M. 2009. A general framework for adaptive and online detection of web attacks. In Proceedings of the 18th international Conference on World Wide Web (WWW '09, poster). Madrid, Spain, April 20 - 24, 2009.
- Goverdhan Singh, Florent Masseglia, Céline Fiot, Alice Marascu and Pascal Poncelet. « Mining Common Outliers for Intrusion Detection ». Advances in Knowledge Discovery and Management (AKDM09), post-actes EGC'09. Accepté sous réserve de modifications. A paraître.
- Nischal Verma, François Trousset, Pascal Poncelet and Florent Masseglia. « Intrusion Detections in Collaborative Organizations by Preserving Privacy ». Advances in Knowledge Discovery and Management (AKDM09), post-actes EGC'09. A paraître.
- Nischal Verma, François Trousset, Pascal Poncelet, Florent Masseglia : « Détection d'intrusions dans un environnement collaboratif sécurisé ».

In Extraction et gestion des connaissances (EGC 2009), Strasbourg, January 2009. pp 301-312. (Nominé pour le prix du meilleur papier applicatif).
- Goverdhan Singh, Florent Masseglia, Celine Fiot, Alice Marascu, Pascal Poncelet : « Collaborative Outlier Mining for Intrusion Detection ». In Extraction et gestion des connaissances (EGC 2009), Strasbourg, January 2009. pp 313-324. (Nominé pour le prix du meilleur papier applicatif).
- Alice Marascu, Florent Masseglia : « Détection d'enregistrements atypiques dans un flot de données : une approche multi-résolution ». In Extraction et gestion des connaissances (EGC 2009), Strasbourg, January 2009. pp 455-456.
- Wei Wang, Thomas Guyet, Rene Quiniou, Marie-Odile Cordier, Florent Masseglia : « Online and adaptive anomaly Detection : detecting intrusions in unlabelled audit data streams ». In Extraction et gestion des connaissances (EGC 2009), Strasbourg, January 2009. pp 457-458.

5.2 Définitions

Clustering, détection d'outliers et fiabilité des alarmes.

Le clustering (ou classification non supervisée) a pour objectif de grouper les objets en fonction de leurs similitudes. Les objets similaires seront placés dans les mêmes clusters (groupes) et les objets dissimilaires seront placés dans des clusters différents. L'évaluation des résultats du clustering porte ensuite sur les similitudes intra-classe et inter-classes.

Les outliers sont généralement considérés comme des objets particulièrement différents des autres. Leur recherche peut se faire par des méthodes basées sur :
- La distance : « il existe un pourcentage d'objets supérieur à une valeur minimum p dont la distance est supérieure à une valeur minimum d ».
- La déviation : « il existe un objet dans le groupe dont les caractéristiques dévient de la description générale de ce groupe ».
- etc.

Dans les section suivantes, nous considérons qu'un flux de données $F = \{P_1, ..., P_i, ..., P_n\}$ est une série de paquets de données P_i, lus par ordre d'arrivée (indice i croissant). Chaque paquet est un ensemble d'objets $O = \{o_1, ..., o_m\}$.

Nous proposons de traiter un flux de données d'usage d'un site Web, par paquet. Soit W le site Web en cours d'analyse. Pour chaque batch, notre but est de définir les clusters et de détecter les événements atypiques parmi les requêtes envoyées aux scripts PHP sur W. Ainsi, nos objets sont les paramètres donnés aux scripts PHP sur W.

Notre objectif est de mesurer la pertinence des outliers détectés afin de déclencher des alarmes. Nous mesurons, dans nos expérimentations, le taux de fausses alarmes selon les définitions classiques du domaine (en considérant C l'ensemble des alarmes qui doivent être déclenchées) :

- Un **vrai positif** est un objet classé dans C et qui appartient à C (soit VP leur nombre).
- Un **vrai négatif** est un objet qui n'est pas classé dans C et qui n'appartient pas à C (soit VN leur nombre).
- Un **faux positif** est un objet classé dans C et qui n'appartient pas à C (soit FP leur nombre).
- Un **faux négatif** est un objet qui n'est pas classé dans C et qui appartient à C (soit FN leur nombre).
- Le **taux de vrais** positifs vaut $\frac{VP}{VP+FN}$.
- Le **taux de faux** positifs vaut $\frac{FP}{FP+VN}$.
- Le **rappel** est égal au taux de vrais positifs.
- La **précision** vaut $\frac{VP}{VP+FP}$.

La précision exprime la pertinence des alarmes d'un IDS (une précision de 100% indique que toutes les alarmes déclenchées correspondent à des intrusions). Le rappel exprime sa capacité à détecter les intrusion (un rappel de 100% indique que toutes les intrusions sont détectées). Bien entendu, le rappel et la précision se situent sur les deux plateaux opposés de la balance. Un rappel de 100% s'obtient facilement en décidant que tout comportement doit déclencher une alarme. Mais dans ce cas, la précision sera la plus faible qu'on puisse obtenir.

5.3 Travaux existants

Les IDS basés sur les anomalies [EAP+02] peuvent être divisés en plusieurs catégories, dont : les supervisés, les semi-supervisés et les non supervisés. La méthode supervisée retient une liste de comportements interdits

et déclenche une alarme quand l'un d'eux est détecté. La méthode semi-supervisée consiste à construire un modèle des comportements normaux du système à partir de données labellisées. Chaque comportement ne correspondant pas à ce modèle sera considéré comme une anomalie et déclenchera une alarme. La méthode non supervisée n'utilise pas de données labellisée. Habituellement, en se basant sur un algorithme de clustering, cette méthode essaie de détecter les outliers et les considère comme des anomalies. La détection d'outlier a été étudiée de façon intensive ces dernières années en raison de ses nombreuses applications, telles que la détection de fraudes à la carte bancaire [AFR97], la cyber sécurité [EEL+04] ou encore la sécurité des systèmes critiques [FYM05]. Ces domaines d'application reposent sur des méthodes de recherche de motifs qui dévient de manière significative d'une notion bien admise de normalité. Le concept d'atypicité a été étudié par les statistiques [MS03, KH07] où les approches construisent des modèles de distribution probabilistes dans lesquels les outliers ont une probabilité faible [BHV00, LX01]. Dans le contexte de la détection d'intrusion, la dimensionalité des données est élevée. Ainsi, pour améliorer les performances globales et maintenir la précision, il est devenu nécessaire de proposer des algorithmes de fouille de données utilisant l'ensemble de la distribution des données ainsi que la plupart des caractéristiques des données [KN98, AY01]. Notre objectif est de proposer un algorithme de détection d'outlier basé sur les résultats d'une segmentation [KN98, RRS00, CAMSC05, FZFW06, JTH01, EAP+02, HXD03, PKGF03]. De telles techniques reposent sur l'idée que les objets normaux appartiennent à des clusters de grande taille alors que les objets atypiques appartiennent à des petits clusters [JTS01, SZ02, FZFW06], ou restent isolés [KN98, RRS00]. En d'autres termes, la détection d'outliers consiste à identifier les objets qui sont isolés des clusters les plus significatifs. Selon l'approche choisie, le nombre de paramètres à donner peut être élevé et influencera le nombre et la qualité des outliers. Pour éviter cela, certaines approches extraient une liste ordonnée d'outliers et limitent le nombre de paramètres [RRS00, JTH01, FZFW06]. Notons que [FZFW06] propose de réduire (ou supprimer) les paramètres de l'algorithme de segmentation, tout en maintenant un paramètre, n, correspondant au nombre d'outliers à extraire. Nous voulons détecter les outliers sur la seule base de leurs caractéristiques. Parmi ces caractéristiques, nous avons sélectionné la taille des clusters. Une distribution sur cette taille, combinée avec notre approche par ondelettes, permet de séparer les clusters en deux sous-ensembles qui correspondent aux petits clusters et aux clusters normaux. Cette approche peut également fonctionner avec n'importe quelle autre caractéristique, telle que la densité des clusters ou le voisinage par exemple.

De toute évidence, les IDS non supervisés souffrent d'un grand nombre de fausses alarmes dans la mesure ou tout nouveau type de comportement sera considéré comme anormal. Les techniques basées sur les signatures de mauvais usages (misuse) feront preuve d'une grande précision mais elles seront aveugles au nouvelles attaques (qui, par définition, ne sont pas dans la liste des attaques connues). Ainsi, on trouve dans la littérature des travaux faisant appel aux techniques collaboratives pour réduire les taux de faux positifs et de faux négatifs [VS01, YBJ04]. Ces approches se basent sur des IDS ditribué gérant une liste noire d'adresses IP construite suite aux comportement malveillants avérés. Si ces communications peuvent conduire à des résultats plus précis, elles ne permettent pas à l'IDS de mettre à jour de nouvelles formes d'attaques et d'éviter les faux positifs.

5.4 Mutualisation des anomalies

Cette section est basée sur l'article « Data Mining for Intrusion Detection : From Outliers to True Intrusions » (Goverdhan Singh, Florent Masseglia, Céline Fiot, Alice Marascu, Pascal Poncelet) publié dans les actes de la conférence internationale PAKDD 2009.

Notre objectif est de réduire le nombre de fausses alarmes des IDS non-supervisés. La principale idée, dans notre approche, est de faire collaborer différents partenaires en exploitant le fait qu'ils partagent les mêmes systèmes (le serveur Web peut être Apache ou IIS, le serveur de données peut tourner sous Oracle, les scripts accédant aux données peuvent être écrits en PHP, ou en CGI, etc.). Quand une faille de sécurité est trouvée sur un système (disons, un script PHP permettant d'obtenir des accès privilégiés avec des paramètres bien précis) alors cette faille sera la même sur tous les partenaires utilisant cette technologie. Notre objectif est d'utiliser cette observation pour réduire le nombre de fausses alarmes.

5.4.1 Principe général

Nous proposons DOC un algorithme permettant de detecter les outliers partagés par au moins deux sites partenaires dans un IDS collaboratif. Par soucis de clarté, nous illustrons notre méthode sur deux sites, S_1 et S_2 et nous considérons les requêtes effectuées sur les scripts de chacun de ces sites (CGI, PHP, SQL, etc.). Notre but est d'obtenir des clusters d'usage sur chaque site et de trouver les outliers communs. De plus, nous souhaitons proposer une méthode ne nécessitant qu'un seul paramètre : n, la limite haute des alarmes renvoyées. En effet, notre algorithme effectue plusieurs étapes de classification pour chaque site. À chaque étape, nous vérifions le nombre d'outliers communs entre les sites. L'algorithme de classification est agglomératif et dépend d'un seuil de dissimilitude maximum (MD) à respecter entre deux objets avant de les regrouper dans un cluster. La figure 5.1 illustre notre approche avec un nombre d'alarmes maximum égal à 1 et une attaque correspondant au cluster A sur les site 1 et 2. À la première étape, la valeur de MD est initialisée avec une valeur très faible afin d'obtenir des clusters aussi petit et dense que possible. Ensuite, nous vérifions les correspondances entre les outliers des sites 1 et 2. Avec les résultats présentés par la figure 5.1 nous obtenons deux correspondances (les outliers A et B) ce qui génère 2 alarmes et dépasse la valeur fixée pour le nombre maximum d'alarmes. Nous augmentons alors la valeur de MD afin d'augmenter la taille

des cluster et de réduire le nombre d'outliers. Après quelques itérations, le processus arrive à l'étape n de la figure 5.1. Le seul outlier commun qux deux sites est alors A, qui correspond à l'attaque et ne génère qu'une seule alarme. Le processus cesse sans aller aux étapes suivantes (m) et sans continuer à faire grandir la valeur de MD.

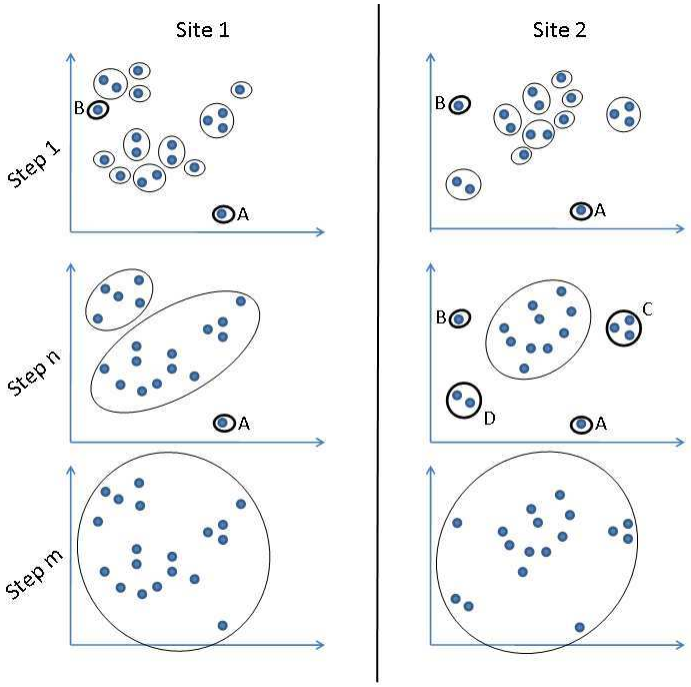

FIG. 5.1 – Detection des outliers communs dans les usages de deux sites

Notre choix de faire varier la valeur de MD en fonction d'un paramètre utilisateur (maximisant le nombre d'alarmes désiré) repose sur l'exploitation de cette méthode dans un contexte temps réel. Dans un tel contexte, les alarmes seraient déclenchées à intervalles réguliers (par exemple chaque heure). En fonction du nombre de fausses alarmes relevées dans le résultat à l'instant t, l'utilisateur peut décider de baisser le nombre d'alarmes (il y a trop de fausses alarmes) ou d'augmenter le nombre d'alarmes (il n'y a que des vraies alarmes et donc on en ignore peut-être).

Ce travail doit donc répondre aux questions suivantes : quelle mesure de similitude utliser entre les données d'usage ? Comment séparer les outliers du reste des clusters ? Comment détecter les outliers communs ? Et surtout,

comment utiliser un seul paramètre dans tout le processus ?

Nous répondons au problème de la détection des outliers dans la section 5.5, avec une approche sans paramètre. Nous traitons les autres problèmes dans la suite de cette section.

5.4.2 DOC : Détection d'Outliers Communs

Le principe de Doc repose sur des étapes successives de classification des usages venant de différents sites partenaires, jusqu'à ce que le nombre d'outliers commun corresponde au nombre d'alarmes désiré. Nous présentons dans ce travail un algorithme conçu pour deux sites seulement. Étendre cet algorithme à plus de deux sites demanderait un serveur central pour coordonner les comparaisons et déclencher les alarmes, ou bien un protocole de communication et de calcul pair-à-pair. Ce n'est pas notre objectif principal. Nous voulons montrer l'intérêt de l'utilisation des outliers communs à plusieurs partenaires dans la détection d'intrusion. Nous travaillerons sur des objets correspondant aux paramètres données par les utilisateurs aux scripts présents sur des sites Web. En d'autres termes, le fichier log d'accès de chaque site est filtré et nous ne gardons que les lignes correspondant à une requête avec paramètre(s) à un script. Pour chaque ligne de ce type, nous isolons les paramètres et à chaque paramètre nous affectons un nouvel objet. Considérons, par exemple, la requête suivante : `staff.php?FName=Jean&LName=Dupont`. Les objets crées suite à cette requête seront $o_1 =$ `Jean` et $o_2 =$ `Dupont`. Une fois les objets crées à partir des usages de chaque site, Doc est appliqué et donne les outliers communs.

Algorithme

L'algorithme Doc est décrit par la figure 5.2. Comme expliqué en section 5.4.1, Doc traite les usages de deux sites, étape par étape. À chaque étape, un résultat de classification est obtenu est analysé. D'abord, la valeur de MD est faible afin d'obtenir des clusters nombreux et petits, puis cette valeur est augmentée par pas de $0,05$ afin d'augmenter la taille des clusters, de diminuer leur nombre et de diminuer le nombre d'alarmes. Quand le nombre d'alarmes désiré est atteint, alors Doc prend fin.

Classification

L'algorithme de clustering utilisé dans Doc est basé sur un principe agglomératif. Il est décrit par la figure 5.3. Le but est d'incrémenter le volume des clusters en ajoutant des objets candidats, jusqu'à ce que le degré de

Algorithm Doc
Input : U_1 et U_2 les usages collectés sur les sites S_1 et S_2
et n la limite haute pour le nombre d'alarmes.
Output : I l'ensemble des clusters correspondants
aux intrusions supposées.

1. $MD \leftarrow 0$;
2. $MD \leftarrow MD + 0.05$;
3. $C_1 \leftarrow Clustering(U_1, MD)$;
   $C_2 \leftarrow Clustering(U_2, MD)$;
4. $O_1 \leftarrow Outliers(C_1)$; $O_2 \leftarrow Outliers(C_2)$;
5. $I \leftarrow CommonOutliers(O_1, O_2, MD)$;
6. If $|I| \leq n$ then return I ;
7. If $MD = 1$ then return I ; // Pas d'outliers communs
8. Else return to step 2 ;

End algorithm Doc

FIG. 5.2 – Algorithme Doc

dissimilitude maximum MD ne soit plus respecté (*i.e.* il y a un objet o_i dans le cluster tel que la dissimilitude entre o_i et l'objet candidat o_c est plus grand que MD).

Mesure de similitude entre les objets. Nous considérons chaque objet comme une séquence de caractères. Notre mesure de dissimilitude (voir la définition 17) est alors basée sur la plus longue sous-séquence commune (LCS) entre les chaînes de caractères représentant les deux objets.

Exemple 9 *Considérons deux paramètres* p_1=`intrusion` *et* p_2=`induction`. *La LCS entre* p_1 *et* p_2 *est* L=`inuion`. *La longueur de* L *est 6 et la dissimilitude entre* p_1 *et* p_2 *est* $d = 1 - \frac{2 \times L}{|p_1|+|p_2|} = 33,33\%$, *ce qui signifie également une similitude de* $66,66\%$ *entre les deux paramètres.*

Centre des clusters. Quand un objet est inséré dans un cluster nous maintennon le centre de ce cluster, qui sera utilisé par l'algorithme CommonOutliers. Le centre C_c d'un cluster C est donné par la LCS calculée sur tous les objets de C. Quand un objet o_i est ajouté à C, la nouvelle valeur de C_c est donnée par la LCS entre C_c et o_i.

Algorithm Clustering
Input : U, les données d'usage et MD, la dissimilitude maximum.
Output : C, l'ensemble des clusters les plus grands possible, respectant MD.

1. Construire M, la matrice des distances entre chaque objet de U ;
2. $\forall p \in M, Neighbours_p \leftarrow$ la liste trié des voisins de p par ordre de similitude décroissante.
3. $DensityList \leftarrow$ liste des objets triés par densité ;
4. $i \leftarrow 0$; $C \leftarrow \emptyset$;
5. $p \leftarrow$ prochain objet non classé de $DensityList$;
6. $i++$; $c_i \leftarrow p$;
7. $C \leftarrow C + c_i$;
8. $q \leftarrow$ prochain objet non classé de $Neighbours_p$;
9. $\forall o \in c_i$, If $distance(o, q) > MD$ then return to step 5 ;
10. add q to c_i ;
11. $C_c \leftarrow LCS(C_c, q)$; $//C_c$ est le centre de C
12. return to step 8 ;
13. Si il reste des objets non classés alors reprendre à l'étape 5 ;
14. return C ;

End algorithm Clustering

FIG. 5.3 – Algorithm Clustering

Comparaison entre les outliers

Nous avons fixé pour contrainte à cette méthode de ne prendre qu'un seul paramètre (la limite haute du nombre d'alarmes). Nous voulons donc éviter d'utiliser un dégré de similitude entre les outliers lors de leur comparaison. L'algorithme CommonOutliers (figure 5.4) va comparer les centres des outliers. Pour chaque paire d'outliers, si la similitude entre les centre est sous le seuil MD alors nous considérons que ces outliers sont similaires et peuvent être ajoutés à la liste des alarmes.

Algorithm CommonOutliers
Input : O_1 et O_2, deux listes d'outliers et MD, la dissimilitude maximum.
Output : A, la liste des alarmes (outliers communs).

1. $A \leftarrow \emptyset$
2. $\forall i \in O_1$ do
3. $\quad \forall j \in O_j$ do
4. $\quad\quad centre_i \leftarrow centre(i)$;
5. $\quad\quad centre_j \leftarrow centre(j)$;
6. $\quad\quad$ If $d(centre_i, centre_j) < MD$
   $\quad\quad\quad$ Then $A \leftarrow A + i \cup j$;
7. $\quad$ done ;
8. done ;
9. Return A ;

End algorithm CommonOutliers

FIG. 5.4 – Algorithme CommonOutliers

5.4.3 Expérimentations

Dans cette section nous analysons le nombre d'outliers et de véritables comportements malicieux et nous indiquons quelques uns de ces comportements. Nos données sont issus des sites Web de l'Inria Sophia-Antipolis et de l'IRISA. Ces données sont analysées sur le mois de mars 2008. Le premier log (Inria Sophia) représente 1,8 Go de données brutes. Dans ce premier fichier, le nombre total d'objets (*i.e.* les paramètres donnés à des scripts) est de 30 454. Le second fichier représente 1,2 Go de données brutes et le nombre total d'objets est de 72 381. DOC a été écrit en Java et C++ sur un PC (2.33GHz i686) exploité par Linux avec 4 Go de mémoire vive. Les paramètres générés automatiquement par les scripts sont supprimés des jeux de données dans la mesure où ils ne peuvent pas correspondre à des attaques (par exemple « *publications.php ?Category=Books* »). Cela peut être fait en listant toutes les possibilités de générer es paramètres automatiquement dans les scripts du site.

Détection d'outliers communs

Doc procède étape par étape en augmentant la valeur de MD permettant de grouper des objets pendant le processus de classification. Dans nos expérimentations, MD a été augmenté par pas de $0,05$ de $0,05$ à $0,5$. Pour chaque pas, nous reportons les mesures dans la table 5.1. La signification de chaque mesure est la suivante : O_1 (resp. O_2) est le nombre d'outliers dans le site 1 (resp. site 2). $\%_1$ (resp $\%_2$) est le rapport entre les outliers et le nombre d'objets sur site 1 (resp. site 2). Par exemple, quand MD vaut $0,3$, pour le site 1 on obtient 5 607 outliers, ce qui représente $18,4\%$ du nombre total des objets (*i.e.* 30 454) du site 1. OC est le nombre d'outliers communs entre les deux sites et $\%_{FA}$ est le pourcentage de fausses alarmes dans les outliers. Par exemple, quand MD vaut $0,05$, on trouve 101 alarmes dont 5 sont fausses (ce qui représente $4,9\%$). Une première observation est que les outliers ne peuvent pas être utilisés directement pour déclencher des alarmes. De toute évidence, un nombre aussi élevé que 5 607 alarmes à vérifier, pour un mois d'usages, n'est pas réaliste. D'un autre côté, les résultats de Doc montrent sa capacité à séparer une catégorie des comportements malicieux et les usages normaux. Nos fausses alarmes correspondent aux requêtes acceptables qui sont communes à deux sites mais peu fréquentes. Par exemple, sur un script d'interrogation des publications d'une équipe de l'Inria Sophia, un utilisateur peut demander les articles de « Jean Dupont » et la requête sera de la forme `publications.php?FName=Jean\&LName=Dupont`. Si un autre utilisateur demande les articles de « Jean Dupuis » sur le site de l'IRISA, la requête sera de la forme `publications.php?FName=Jean\&LName=Dupuis` et le paramètre « Jean » sera détecté comme outlier commun. Toutefois, comme on peut le constater dans nos résultats, le taux de fausses alarmes $\%_{FA}$ est très faible (généralement, nous obtenons 5 fausses alarmes au maximum dans nos expérimentations sur les deux sites Web) en comparaisons des centaines d'outliers qui sont filtrés par Doc. On peut également observer qu'une valeur faible de MD génère de petits clusters et de nombreux outliers. Certains de ces outliers sont partagés entre les deux sites alors qu'ils représentent des comportements normaux mais rares. Puis, quand la valeur de MD augmente, le processus de clustering devient plus agglomératif et les alarmes sont groupées. En même temps, le nombre d'outliers correspondant à des usages normaux diminue (car ils sont également groupés). Finalement, une valeur de MD trop grande génère des clusters qui n'ont plus vraiment de sens. Dans ce cas, les outliers deviennent plus grands et les critères de correspondances sont plus tolérants ce qui entraine un grand nombre d'outliers contenant des usages normaux. Si Doc était utilisé dans un environnement de flux avec ces

	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
O_1	13197	10860	8839	7714	6547	5607	5184	4410	3945	3532
$\%_1$	43.3%	35.6%	29%	25.3%	21.5%	18.4%	17%	14.4%	12.9%	11.6%
O_2	35983	27519	24032	20948	18152	14664	12738	11680	10179	8734
$\%_2$	49.6%	37.9%	33.1%	28.9%	25%	20.2%	17.5%	16.1%	14%	12.1%
OC	101	78	74	70	67	71	71	85	89	90
$\%_{FA}$	4.9%	5.12%	4%	2.85%	1.5%	2.8%	2.8%	10.6%	11.2%	16.6%

TAB. 5.1 – Resultats sur les données Inria et IRISA

données, l'utilisateur pourrait choisir un nombre de 70 alarmes et surveiller le taux de fausses alarmes. Si ce taux diminue, alors l'utilisateur peut choisir d'augmenter le nombre d'alarmes, et vice-versa (afin de minimiser le nombre de faux négatifs).

Quelques alarmes selon DOC

En mars 2008, le site Web de l'Inria Sophia-Antipolis a été victime d'une attaque utilisant le logiciel bibAdmin, basé sur PHP et permettant de gérer les publications d'une équipe de recherche. La faille consistait à ajouter un fichier avec l'extension .bib même si il ne s'agissait pas d'un fichier bibTex. Le contenu de ce fichier était alors déposé sur le site et pouvait être lu depuis l'extérieur. Si le contenu du fichier était un ensemble de commandes exécutables, alors elle pouvaient être lancées depuis un navigateur extérieur en appelant ce fichier ultérieurement. Cette attaque a provoqué la défiguration momentanée du site de Sophia et une enquête de plusieurs semaines pour remonter à la source de cette faille et au mode opératoire. D'un autre côté, cette attaque avait également eu lieu à l'IRISA quelques jours avant (sans succès). Si une approche du type DOC avait été mise en place sur les deux sites en Mars 2008, la première tentative aurait été détectée sur le site de l'IRISA sans déclencher d'alarme (car l'anomalie n'aurait pas été partagée, étant la première tentative). En revanche, la tentative sur le site de Sophia aurait été détectée car elle aurait eu les mêmes paramètres que celle effectuée sur le site de l'IRISA (l'anomalie commune aurait été détectée). Nos expérimentations ont montré que le paramètre `ref=add.php` était la signature commune pour cette attaque, sur les deux sites et que DOC était en mesure de détecter cette attaque en se basant sur les anomalies communes.

Les attaques trouvées dans la suite de ce texte ont toutes échoué, mais leur détection montre le bien-fondé de notre approche.

– **Injection de code :** un type récent d'attaque consiste à injecter du

code PHP dans des scripts en donnant une URL en paramètre. Quand le code à faire exécuter par le script victime est trop long (taille maximum de paramètre dépassée) cette technique permet de contourner la limite et de passer le code tout de même. Voilà un échantillon des URL ainsi détectées : http ://myweddingphotos.by.ru/images ? http ://levis-potparty.eclub.lv/images ? http ://0xg3458.hub.io/pb.php ? Selon les réglages PHP sur le site victime, le code injecté peut permettre la modification du site. Ces URL sont données en paramètres de manière automatique et massive à des scripts via des batch d'instructions (statistiquement, l'attaquant espère trouver un site vulnérable).

- **Mots de passe :** un autre genre (naïf) d'attaque consiste à tenter de récupérer le fichier des mots de passe d'un système, via son site Web. Les tentatives sont alors caractérisées par des paramètres du type ../etc/passwd avec un nombre variable de « ../ » en début de paramètre. Cette tentative est probablement la plus courante. Elle n'est généralement pas dangereuse (car bien connue) mais doit être détectée pour montrer l'efficacité d'un outil de détection d'intrusions.
- **Les œufs de pâques :** il ne s'agit pas vraiment d'une attaque, mais les caractéristiques sont strictement les mêmes. Sur la plupart des serveurs, si on ajoute le code suivant : ?=PHPE9568F36-D428-11d2-A769-00AA001ACF42 à la fin de n'importe quelle page PHP, un affichage se produit avec des images des développeurs. De plus, le 1er avril, une image remplacera le logo PHP sur la page phpinfo(). Ce code est détecté comme une anomalie commune car il ne concerne pas les données gérées par le site, mais bien une particularité du code utilisé par plusieurs sites.

5.5 Détection d'anomalies par les ondelettes

Cette section est basée sur l'article « Parameterless outlier detection in data streams » (Alice Marascu, Florent Masseglia) publié dans les actes de la conférence internationale ACM SAC 2009 (Data Stream Track).

À notre connaissance, la détection d'outlier repose toujours sur un paramètre [JTH01, ZKS07, PES01, JORL04], tel qu'un pourcentage sélectionné dans les petits clusters ou encore les top-n outliers. Généralement, l'idée consiste à trier les clusters ou les objets selon un critère défini (taille du cluster, densité des objets ou des clusters, ...). Nous considérons que nos clusters sont aussi denses que possible, grâce à notre algorithme de segmentation, et nous voulons extraire les outliers en utilisant le critère de la taille (faisant l'hypothèse que les événements atypiques sont moins nombreux que les événements normaux). Le problème consiste donc à séparer les clusters, en fonction de leur taille, dans deux catégories (les normaux et les outliers). Notre solution se base sur une analyse de la distribution des clusters, après les avoir triés par taille croissante. Une distribution classique est illustrée par la figure 5.5 (capture d'écran réalisée avec nos données réelles). L'idée de Doo est d'utiliser la transformée en ondelettes de cette distribution pour trouver la meilleure séparation.

5.5.1 Détection d'outliers non-paramétrée

Grâce à une connaissance *a priori* sur le nombre de plateaux (on veut deux plateaux, un pour les petits segments, ou outliers, et un pour les segments normaux) nous pouvons couper cette distribution d'une manière efficace. Dans la figure 5.5, la taille des clusters est reportée en y et leur indice dans la distribution est reporté en x. Les deux plateaux correspondent à la séparation entre les outliers et les clusters normaux. En effet, chaque cluster dont la taille est inférieure ou egale à la valeur du premier plateau sera considéré comme un outlier. La transformée en ondelettes est un outil qui permet de décomposer des données, ou des fonctions, ou des opérateurs, en différentes composantes de fréquence et ensuite d'étudier chaque composante avec une résolution adaptée à son échelle [Dau92]. En d'autres termes, la théorie des ondelettes permet de représenter une série de valeurs et la décompose en plusieurs parties reliées ; quand ces parties sont mises à l'échelle et translatées, cette décomposition est appellée tranformation en ondelettes. La reconstruction en ondelettes, ou transformée en ondelettes inverse, implique de remettre les différentes parties ensemble et de retrouver l'objet

H

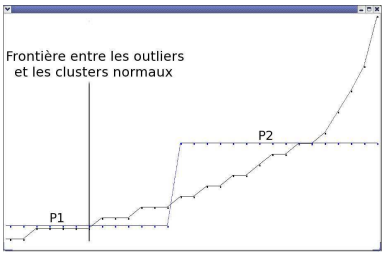

FIG. 5.5 – Détection d'outliers par les ondelettes de Haar

d'origine [You95].

Du point de vue mathématique, la transformée en ondelettes continue est définie par :

$$T^{wav}f(a,b) = \frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty} f(x)\psi^*(\frac{x-b}{a})dx$$

où z^* dénote le nombre complexe conjugué de z, $\psi^*(x)$ est l'ondelette, a (> 0) est le facteur de mise à l'échelle et b est le paramètre de translation. Cette transformation est linéaire et co-variante en translations et dilatations. Cette expression peut être également interprétée comme une projection du signal sur une famille de fonctions analysant $\psi_{a,b}$, construite à partir d'une ondelette mère et selon l'équation suivante : $\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$. Les ondelettes sont une famille de fonctions localisées en temps et en fréquence et sont obtenues par translations et dilatations à partir d'une fonction $\psi(t)$, dite ondelette mère. Pour des choix particuliers de a, b et ψ, $\psi_{a,b}$ est une base orthonormale pour $L^2(\mathbb{R})$. Tout signal peut être décomposé en le projetant sur sa fonction d'ondelette. Pour comprendre le mécanisme de la transformée en ondelettes, il faut comprendre l'analyse multi-résolution (AMR). Une analyse multi-résolution de l'espace $L^2(\mathbb{R})$ est une séquence de sous-espaces imbriqués tels que :

$... \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} ... \subset V_{j+1} \subset V_j ...$

$$\overline{\bigcup_{j\in Z} V_j} = L^2(\mathbb{R})$$
$$\bigcap_{j\in Z} V_j = \{0\}$$

$\forall j \in \mathbb{Z}$ if $f(x) \in V_j \Longleftrightarrow f(2^{-1}x) \in V_{j+1}$ (or $f(2^j x) \in V_0$)

$\forall k \in \mathbb{Z}$ if $f(x) \in V_0 \Longleftrightarrow f(x-k) \in V_0$

Il existe une fonction $\varphi(x) \in L^2(\mathbb{R})$, appellée fonction de mise l'échelle

qui, par dilatation et translation, génère une base orthonormale de V_j. Les fonctions sont construites selon la relation suivante : $\varphi_{j,n}(x) = 2^{-\frac{j}{2}}\varphi(2^{-j}x - n), n \in \mathbb{Z}$, et la base est orthonormée si $\int_{-\infty}^{+\infty} \varphi(x)\varphi^*(x+n)dx = \delta(n), n \in \mathbb{Z}$. Pour chaque V_j, son complément orthogonal W_j dans V_{j-1} peut être défini comme suit : $V_{j-1} = V_j \oplus W_j$ et $L^2(\mathbb{R}) = \bigoplus_{j\in\mathbb{Z}} W_j$. Comme W_j est orthogonal à V_{j-1}, alors W_{j-1} est orthogonal à W_j, donc $\forall j, k \neq j$ on a $W_j \perp W_k$.

Il existe une fonction $\psi(x) \in \mathbb{R}$, appellée ondelette qui, par dilatations et translations, génère une base orthonormale de W_j, et donc de $L^2(\mathbb{R})$. Les fonctions sont construites comme suit : $\psi_{j,n}(x) = 2^{-\frac{j}{2}}\psi(2^{-j}x - n), n \in \mathbb{Z}$. Donc, $L^2(\mathbb{R})$ est décomposé en une séquence infinie d'espaces d'ondelettes, *i.e.* $L^2(\mathbb{R}) = \bigoplus_{j\in Z} W_j$. Pour résumer la décomposition en ondelettes : soit une fonction f_n dans V_n, f_n est décomposée en deux parties, une partie dans V_{n-1} et l'autre dans W_{n-1}. À l'étape suivante, la partie V_{n-1} est encore décomposée en deux parties, une partie dans V_{n-2}, l'autre dans W_{n-2} et ainsi de suite...

Une application directe de l'AMR est la transformée rapide discrete en ondelettes (Discrete Wavelet Transform). L'idée est d'aplanir les données de manière itérative et de garder les détails séparément. Des preuves plus formelles sur les ondelettes peuvent être trouvées dans [Dau92]. Les ondelettes sont généralement utilisées pour résumer des données ou trouver une tendance dans des fonctions numériques. En pratique, la majorité des coefficients d'ondelettes sont petits ou insignifiants. Ainsi, pour représenter une tendance, seul un petit nombre de coefficients significatifs est nécessaire. Nous utilisons les ondelettes de Haar dans notre méthode de détection d'outliers. Considérons la série de valeurs suivante : $[1, 1, 2, 5, 9, 10, 13, 15]$. Sa transformée en ondelettes de Haar est illustrée dans la table suivante :

Niveau	Approximations	Coefficients
8	1, 1, 2, 5, 9, 10, 13, 15	
4	1, 3.5, 9.5, 14	0, -1.5, -0.5, -1
2	2.25, 11.75	-1.25, -2.25
1	7	-4.75

On garde alors les deux coefficients les plus significatifs (voir la proposition 1) et les autres sont mis à zéro. Dans notre série de coefficients ([7, $-4,75$, -1.25, -2.25, 0, -1.5, -0.5, -1]) les deux plus significatifs sont 7 et -4.75, afin que la série deviennne [7, -4.75, 0, 0, 0, 0, 0, 0]. Dans les étapes suivantes, la transformée inverse est calculée et nous obtenons

une approximation des données d'origine [2.25, 2.25, 2.25, 2.25, 11.75, 11.75, 11.75, 11.75]. Cette approximation donne deux plateaux, correspondants aux valeurs $\{1, 1, 2, 5\}$ et $\{9, 10, 13, 15\}$. Dans notre cas, l'ensemble des outliers contient les clusters dont la taille est inférieure au premier plateau (*i.e.* 2.25). Dans notre exemple, $o = \{1, 1, 2\}$ donne la taille des outliers (*i.e.* les clusters dont la taille est inférieure ou égale à 2).

Plus généralement, les avantages de cette méthode pour notre problème sont illustrés à la figure 5.6. Selon la distribution, la transformée en ondelettes donne différents indices (où couper). Par exemple, dans nos données d'usage de *Lab_anonyme* il y a une variation de la distribution entre le jour et la nuit sur certains scripts PHP. Cette variation aboutit à deux formes principales. La figure 5.6 donne une illustration de deux distributions différentes, similaires à celle retrouvées dans nos expérimentations. Considérons qu'un filtre de 10% soit appliqué à cette distribution pour en extraire les outliers. Si ce filtre obtient des résultats corrects sur la première distribution (partie gauche correspondant aux requêtes vers 01h00) il est en revanche inadapté à la distribution de droite (usages vers 17h00) et renvoie des clusters qui ne devraient pas être considérés comme outliers. Notre principe de détection d'outliers à base d'ondelettes, par contre, est auto-adaptatif et s'ajustera pour prendre en compte la nouvelle forme de la distribution en augmentant très peu le niveau de sélection des outliers. L'application de la transormée

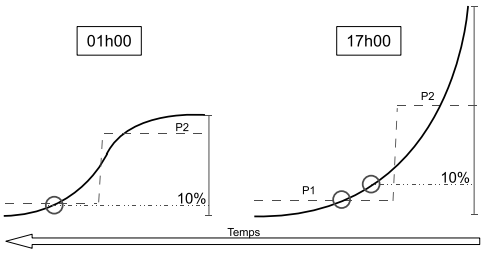

FIG. 5.6 – Une distribution qui varie avec le temps

en ondelettes sur les séries de valeurs permet d'obtenir une bonne compression et, en même temps, selon différentes tendances, une bonne séparation. Sachant que les outliers sont des objets rares, il seront toujours groupés dans les petits clusters (ou resteront isolés). Le principe de DOO, pour séparer les outliers des clusters normaux, est basé sur la proposition 1.

Proposition 1 *Soit F, l'ensemble des caractéristiques utilisées pour construire la distribution D sur les résultats de l'algorithme de segmentation. Soit P1 et P2 les deux plateaux obtenus après avoir sélectionné les deux coefficients les plus significatifs de la transformée en ondelettes de D. La séparation optimale en deux groupes, selon F et respectant la minimisation de la somme des erreurs carrées, est données par P1 et P2.*

Dans une base orthonormale, il est montré que les k coefficients d'ondelette les plus grands donnent la meilleure k-approximation de Haar du signal d'origine, selon la minimisation de la somme des erreurs carrés pour un k donné [SDS95]. Pour cela, considérons le signal d'origine $f(x)$ et la fonction $u1(x)$, $...um(x)$. Le signal peut alors être représenté selon la fonction comme : $f(x) = \sum_{i=1}^{m} c_i u_i(x)$

Le but est de trouver une approximation avec moins de coefficients. Soit σ une permutation de $1, ..., m$ et f' la fonction d'approximation utilisant les premiers m' éléments de σ, avec $m' < m$. $f'(x) = \sum_{i=1}^{m'} c_{\sigma(i)} u_{\sigma(i)}(x)$

La somme des erreurs carrés de L^2 de cette approximation est :

$$
\begin{aligned}
\left|\left|f(x) - f'(x)\right|\right|_2^2 &= < f(x) - f'(x) | f(x) - f'(x) > \\
&= \left\langle \sum_{i=m'+1}^{m} c_{\sigma(i)} u_{\sigma(i)} \Big| \sum_{j=m'+1}^{m} c_{\sigma(j)} u_{\sigma(j)} \right\rangle \\
&= \sum_{i=m'+1}^{m} \sum_{j=m'+1}^{m} c_{\sigma(i)} c_{\sigma(j)} < u_{\sigma(i)} | u_{\sigma(j)} > \\
&= \sum_{i=m'+1}^{m} (c_{\sigma(i)})^2
\end{aligned}
$$

Grâce à l'orthonormalité, $< u_i, u_j >= \delta$, donc pour tout $m' < m$, pour minimiser cette erreur le meilleur choix de σ est la permutation croissante (ou la permutation qui contient les éléments ordonnés par ordre croissant). Donc, pour $m' = 2$ on obtient la meilleure 2-approximation de Haar du signal d'origine.

Sur la base de la proposition 1, on retient les 2 coefficients les plus significatifs et on sélectionne les clusters dont la taille est inférieure ou égale à la valeur du premier plateau. Ces clusters sont alors considérés comme des outliers sans qu'aucun paramètre ne soit demandé à l'utilisateur.

5.5.2 Expérimentations

Dans un environnement de flux de données il est difficile de choisir le bon "degré" d'atypicité. Cette difficulté peut venir du très faible temps disponible pour prendre une décision, ou encore de la variation de ce degré d'un instant à l'autre dans le flux. Dans ces conditions, une méthode de détection des événements atypiques ne doit pas dépendre d'un paramètre tel que k, pour les top-k outliers, ou p, le pourcentage de clusters en queue de distribution. D'un autre côté, une méthode non-paramétrée doit garantir de bons résultats en séparant de manière pertinente les clusters et les outliers. Elle doit également être auto-adaptative et s'ajuster en fonction des changements de distribution (exponentielle, logarithmique, linéaire, etc.). Enfin, elle doit s'adapter à n'importe quelle taille de distribution et de clusters. Ces expérimentations sont destinées à explorer les capacités de DOO à respecter ces caractéristiques.

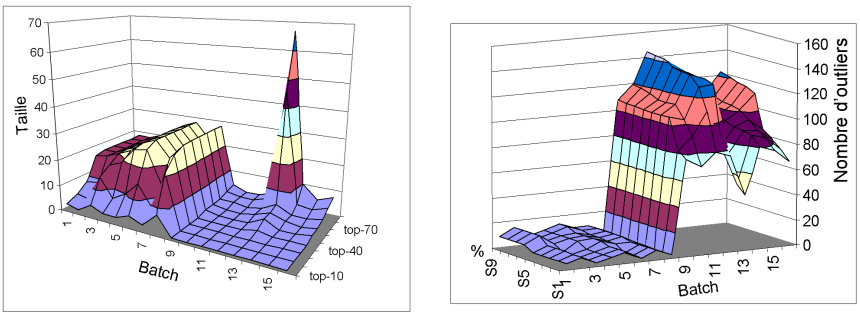

FIG. 5.7 – Taille des outliers avec un filtre top-k et nombre d'outliers avec un filtre p%

Ces expérimentations ont été réalisées sur des données réelles, venant de log d'usage Web de *Lab_ anonyme* de janvier 2006 à avril 2007. Les données originales représentent 18 Go et correspondent à un total de 11 millions de navigations, regroupées par paquets de 8500 requêtes (en moyenne). Dans cette section nous montrons des résultats sur 16 paquets bien représentatifs des résultats globaux et qui illustrent bien les changements de distribution. Les 8 premiers paquets sont sélectionnés sur les requêtes PHP executées entre 01h00 et 02h00. Les 8 paquets suivants, sont sélectionnés entre 15h et 16h.

La figure 5.7 montre les résultats obtenue par un top-k et un p% sur les 16 paquets. Pour chaque paquet, le nombre d'objets et de clusters est donnée par la table 5.2. La première surface de la figure 5.7 (gauche) donne la taille

Paquet	1	2	3	4	5	6	7	8
Objets	1425	1365	1805	1600	1810	2115	1855	2930
Clusters	44	42	35	37	34	34	41	37
Paquet	9	10	11	12	13	14	15	16
Objets	1980	2225	2175	2470	3645	2730	4040	4105
Clusters	109	124	148	133	84	99	120	134

TAB. 5.2 – Paquets, objets et clusters

des clusters sélectionnés par un filtre top-k. Le principe est de sélectionner les k derniers clusters après les avoir triés par taille décroissante. Un désavantage évident est de sélectionner soit trop, soit pas assez, de clusters. Considérons, par exemple, le paquet 13 de la figure 5.7 (gauche). Avec $k = 50$, la taille maximum d'un outlier est de 4, alors qu'avec $k = 90$, cette taille est de 67 (ce qui est la taille maximum d'un cluster dans ce paquet qui n'en contient que 84).

Nous avons également implémenté un filtre basé sur $p\%$, le pourcentage de clusters à considérer comme outliers dans la distribution. Le nombre d'outliers sélectionnés par ce filtre, selon différentes valeurs de p (*i.e.* de 0.01 à 0.09), est donné dans la figure 5.7 (surface de droite). Le principe est de considérer $p \in [0..1]$, un pourcentage donné par l'utilisateur, $d = maxVal - minVal$ l'intervalle des tailles de clusters et $y = (p \times d) + minVal$. Ensuite, le filtre sélectionne les clusters de taille t tels que $t \leq y$. Par exemple, avec $s = \{1, 3, 10, 11, 15, 20, 55, 100\}$, une distribution de tailles et $p = 0.1$, on obtient $d = 100 - 1 = 99$, $y = 1 + (0.1 \times 99) = 10$ et l'ensemble des outliers sera $o = \{1, 3, 10\}$. Dans nos expérimentations, ce filtre est généralement plus résistants aux variations de distribution que le filtre top-k. On peut noter des résultats plus homogènes dans la figure 5.7 (absence de "pics"). Par exemple, avec le paquet 13, on note que ce filtre extrait des outliers de taille 44 (1%) à 78 (9%), ce qui correspond aux résultats des filtres top-40 à top-70.

La figure 5.8 donne une comparaison de DOO (sur les même paquets) avec les filtres top-k et $p\%$. La partie gauche montre les résultats de DOO, d'un top-10 et d'un top-70. Sur les 8 premiers paquets, DOO et le top-10 donnent les meilleurs résultats. Cependant, pour les paquets 9 à 16, le top-10 donne des résultats trop petits (taille maximum de 1). Sur les paquets 9 à 16, les meilleurs résultats sont donnés par DOO et le top-70. En revanche, le top-70 donne des résultats bien trop élevé sur les 8 premiers paquets. En conclusion, aucune valeur de k ne peut servir de référence tout le long du flux et l'utilisateur devra modifier k en fonction des changements de

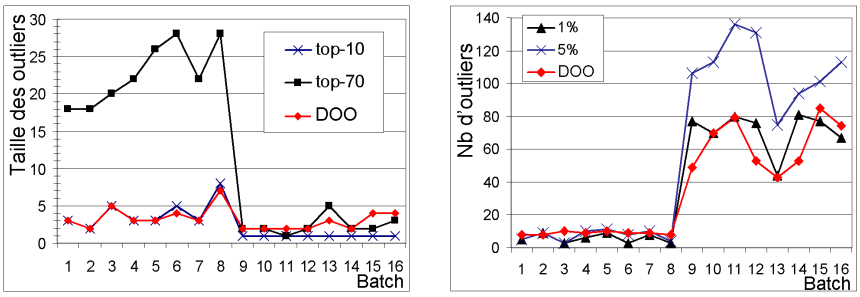

FIG. 5.8 – Comparaison entre DOO et les filtres top-k et p%

distribution d'un paquet à l'aute. Ces résultats font du top-k une méthode difficile à utiliser dans le contexte des flots. D'un autre côté, les résultats de DOO montrent sa capacité à s'adapter d'une distribution à l'autre et à sélectionner automatiquement la bonne taille des événements atypiques.

La partie droite de la figure 5.8, compare les résultats de DOO avec ceux des filtres 1% et 5%. Nous observons que DOO et le filtre 1% donnent des résultats similaires. Par exemple sur le paquet 7, DOO obtient 9 outliers et le filtre 1% en obtient 8. Cependant, la plupart des valeurs du filtre 1% sont basses sur les 8 premiers paquets. Sur ces paquets, le filtre 5% obtient de meilleurs résultats, mais il n'est plus du tout adapté aux paquets 9 à 16 (jusqu'à 138 outliers contre au plus 84 pour DOO). Cela est dû à la variation de distribution, comme illustré par la figure 5.6.

5.6 Discussion

Ce chapitre a présenté une technique de détection d'intrusion basée sur l'idée qu'une faille, après sa découverte, sera exploitée sur plusieurs sites et sera un comportement atypique sur chacun des sites en question. Pour mettre en œuvre la méthode qui exploite cette idée, il fallait proposer 1) un algorithme de clustering non-supervisé capable de grouper les usages similaires et 2) une technique de détection des comportements atypiques non-paramétrique.

Avec la méthode COD nous avons montré que ces comportements atypiques partagés sur plusieurs sites sont la plupart du temps des comportements malicieux. De plus, nous avons vu que les outliers ne peuvent pas être utilisés directement pour déclencher des alarmes en raison de leur trop grand nombre.

Il faut savoir que dans le domaine de la détection d'intrusion, le taux de fausses alarmes est calculé comme étant le nombre d'alarmes sur le nombre total de comportements. Ainsi, avec 2 millions de comportements (ce qui n'est pas rare pour une semaine d'analyse d'un site de grande audience) si le nombre de fausses alarmes atteint 20 000 alors le taux de fausses alarmes sera de 1%. Ce taux, exprimé en pourcentage, peut paraître faible mais il s'agit tout de même d'un montant de 20 000 alarmes traitées alors qu'elles ne correspondent pas à des attaques.

Dans les résultats de COD, le taux de faux positifs est extrêmement faible. Avec une moyenne de 2 fausses alarmes pour 72 381 comportements, le taux de fausses alarmes est d'environ 0,002%.

D'un autre côté, la précision de COD sera plus faible qu'avec un système de détection d'intrusion basé sur les anomalie traditionnel. Cependant, notre objectif est d'apporter une réponse fiable au problème des faux positifs. COD n'a pas vocation à être utilisé tel quel pour constituer un IDS. Il s'agit d'une brique dans le mur des IDS et nous pensons que la solidité de cette brique et la fiabilité de ces résultats sont suffisants pour apporter un élément de réponse à ce problème des faux positifs.

Dans un deuxième temps, nous avons présenté DOO, une méthode de détection d'objets atypiques dans les résultats d'un algorithme de segmentation. DOO ne nécessite aucun réglage. Son principe est de séparer les outliers des clusters dans une distribution calculée selon une ou plusieurs caractéristiques (densité, taille, voisinage,...). Grâce à une décomposition en ondelettes, DOO est capable de proposer cette séparation de manière efficace. Les avantages de DOO sont i) une adaptation automatique à toute forme

de distribution et ii) des résultats pertinents et naturels. Nos expérimentations montrent que DOO présente un double avantage sur deux méthodes reconnues de détection d'outliers (les filtres top-k et pourcentage) :

1. DOO n'a pas besoin de paramètre. Cette méthode s'ajuste automatiquement à toute forme de ditribution et tout nombre d'objets et de clusters. En revanche, l'utilisateur doit essayer plusieurs valeurs de pourcentages avant de trouver le bon intervalle (par exemple 5% pour les premiers paquets). Les outliers donnés par DOO resteront valides après un changement de distribution.
2. Selon le théorème 1, DOO donne une séparation optimale entre les clusters et les outliers. Considérons l'exemple précédent avec la distribution $s = \{1, 3, 10, 11, 15, 20, 55, 100\}$. Avec cette distribution, le filtre 10% donnera l'ensemble d'outliers $o = \{1, 3, 10\}$. Pourquoi ne pas inclure 11 dans o ? En effet, 10 et 11 sont des valeurs très proches. D'un autre côté, DOO donnera l'ensemble $o = \{1, 3\}$, ce qui est un résultat réaliste et naturel.

5.6.1 Confidentialité des échanges dans le cadre de DOC

Dans le monde réel, une application telle que DOC demande évidemment de travailler sur deux points importants, à savoir 1) la gestion de plus de deux partenaires et 2) la confidentialité des données echangées entre les partenaires.

En ce qui concerne le passage à plusiseurs sites partenaires, nous considérons que le fait d'étendre DOC à plus de deux sites demanderait un serveur central pour coordonner les comparaisons et déclencher les alarmes, ou bien un protocole de communication et de calcul pair-à-pair.

Nous avons étudié la confidentialité des échanges entre les sites partenaires. Il s'agit de proposer des protocoles de communication basés sur l'échange de signatures écrites sous forme d'expressions régulières et garantissant qu'aucune information sur le contenu du site n'est dévoilée. Ces travaux ont été acceptés dans l'ouvrage AKDM qui fait suite à la conférence EGC'09 (à paraître).

5.6.2 Méthode à réservoir

Dans [WMG+09], lors du post-doc de Wei Wang, nous avons proposé un autre environnement de détection d'intrusions, basé sur la détection du changement. Il s'agit d'un travail différent, dans la mesure ou l'objectif était

d'obtenir cette détection dans les flux de données. Dans ce travail, nous considèrons également que les comportements malicieux vont constituer des outliers lors d'un processus de classification. Ainsi, la taille et l'isolement des clusters sera utilisée pour mesurer le degré de dangerosité. L'environnement proposé détecte les attaques en trois étapes :

1. Construction du modèle initial grâce à un algorithme de clustering adapté aux flux de données. Une fois les clusters obtenus, des représentants de clusters sont choisis (un représentant est un membre du cluster qui tient lieu de centre). Les outliers sont identifiés, marqués comme suspects et placés dans un réservoir.
2. Identification des outliers et mise à jours du modèle à partir du flux de données d'usages. Chaque comportement arrivant dans le flux est comparé aux représentants. Si aucun représentant n'est assez proche, le comportement est considéré comme suspect et va dans le réservoir. Sinon, le comportement est considéré comme normal et le modèle est mis à jour de manière incrémentale.
3. Reconstruction du modèle et identification des attaques. Le modèle est reconstruit si le nombre d'outliers dépasse un seuil fixé ou si un délai précis est dépassé. Le modèle est reconstruit en utilisant les représentants et les outliers du réservoir en utilisant le même algorithme de clustering qu'à l'initialisation. Si un outlier reste dans le réservoir après cette étape, alors il déclenche une alarme.

Chapitre 6

Conclusion et perspectives

6.1 Conclusion

La fouille de données d'usages est un domaine aux frontières très souples. Cette souplesse est due à l'évolution perpétuelle des caractéristiques du domaine et de ses données. À l'époque de mon doctorat, ces données étaient disponibles dans des quantités qui nous paraissaient presque ingérables. Il s'agissait d'environ 100 mégo-octets par mois. Ces traces d'usages étaient de véritables mines d'or car elles nous permettaient de tester des algorithmes d'extraction de connaissances sur des données réelles et dans des quantités que nous considérions comme très grandes !

Aujourd'hui on parle de flux de données, de rafales, de résumés approximatifs, de périodes, d'attaques distribuées ou encore de comportements tellements variés qu'aucun n'est vraiment fréquent. De mon point de vue, une des caractéristiques des usages du Web aujourd'hui domine les autres : le *dynamisme*. Ce dynamisme entraîne forcément des évolutions. Lors des travaux présentés dans ce mémoire, ce dynamisme était une motivation importante. Le fait que les comportements véritablement fréquents n'existent plus en raison de leur diversité nous a conduit à extraire des motifs avec des supports extrêmement faibles. Le fait que ces comportements apparaissent et disparaissent en suivant l'évolution du système et de l'actualité nous a conduit à découvrir ces périodes d'activités. La rapidité de production de ces traces d'usage ayant dépassé les méthodes développées jusqu'ici pour les analyser nous a conduit à introduire de l'approximation dans l'analyse et la gestion de l'historique des connaissances.

De manière générale, ces travaux sont également guidés par des théories, des questionnements et l'exploration de domaines encore peu ou pas étudiés :

- « Avec un support à la limite inférieure des valeurs acceptables, les fréquents sont encore peu utiles. Baisser le support entraine l'échec des méthodes d'extraction. Pourtant, il existe certainement des motifs pertinents avec des supports plus faibles ». Pour vérifier cette hypothèse, les techniques présentées dans le chapitre 2 ont montré que ces motifs existent, qu'ils sont instructifs dans une analyse des usages et que leur extraction peut être obtenue par une méthode divise ou par la généralisation des pages.
- « On sait découper les données pour en extraire des connaissances sur des périodes *a-priori* intéressantes, mais il en existe probablement d'autres ». Cette hypothèse est très représentative de ce qui motive généralement mes travaux : éviter d'injecter des connaissances *a-priori* dans le processus d'extraction des connaissances. De mon point de vue, la fouille de données doit permettre de découvrir des connaissances sans utiliser d'*a-priori*, comme je le décris en section 1.2.3 et dans la discussion 3.6. Les travaux liés à cette hypothèse sont décrits dans le chapitre 3 avec l'extraction de périodes associées aux motifs fréquents qu'elles contiennent.
- « L'approximation est indispensable pour extraire des connaissances dans des flux de données. Il faut intégrer cette approximation dans l'analyse des flux de données séquentielles ». Ce besoin très pragmatique vient de l'arrivée des flux de données et de leurs caractéristiques sur-contraintes. Il faut simplement être capable de proposer des résultats de fouille de données dans un contexte qui interdit les méthodes traditionnelles. Le chapitre 4 a proposé une technique d'extraction de motifs approximatifs (les séquences alignées) et un nouveau modèle de gestion de l'historique (REGLO) pour les flux.
- « La détection d'anomalies ne peut pas, à elle seule, répondre au problème de la détection d'intrusion car beaucoup d'anomalies ne sont pas dangereuses. Il doit exister une caractéristique aux intrusions qui permettrait de les distinguer des anomalies non dangereuses ». La découverte d'un comportement malicieux parmi les motifs aux supports très faibles a déclenché cette hypothèse. L'explication donnée par nos services informatique à l'Inria Sophia Antipolis nous a incité à chercher des motifs atypiques (première caractéristique) et partagés (seconde caractéristique) sur plusieurs sites. L'hypothèse que ces deux caractéristiques permettent de réduire (et presque annuler) les faux positifs a été vérifiée dans le chapitre 5.

Les travaux présentés dans ce mémoire ont donc été animés par ces hypothèses et par les possibilités d'applications concrètes des résultats envisageables. Ils ont permis de répondre, au moins en partie, à ces théories et de contribuer aux domaines concernés. Ils ont été conduits avec une attention particulière portée sur l'utilité de leurs résultats et le pragmatisme de leurs applications. Les mener à bien demandait donc une combinaison incessante de théorie et de mise en pratique.

6.2 Perspectives

Les travaux présentés dans ce mémoire, ainsi que les travaux que j'ai réalisés de manière générale, ouvrent des perspectives dont certaines sont exposées dans la suite de cette section.

6.2.1 Les transactions ne sont pas atomiques

Les travaux existants en extraction d'itemsets dans les flux ont tous considéré des transactions d'itemsets, c'est à dire des transactions contenant immédiatement plusieurs items pour un client. Plusieurs façons de découper le flux sont ensuite proposées (batches, sliding windows, etc.) sur les modèles présentés en section 4.3. Cependant, les flux du monde réel ne sont pas toujours faits de transactions d'itemsets. Ils sont faits de transactions d'items. Citons, par exemple les données d'usages issues des navigations Web (les clients ne cliquent que sur une seule page à la fois), les transaction de cartes bancaires (une seule utilisation de la CB à la fois), Les achats de places de cinéma (un seul film à la fois), etc. Autrement dit, nous ne trouvons aucun travail proposant l'extraction d'itemsets fréquents sur des enregistrements qui se constituent de manière séquentielle.

Dans le cadre de la thèse de Chongsheng Zhang, nous développons une structure de données permettant d'observer le flux. Cette structure est une file de type First In First Out (FIFO) de taille fixe. Une cellule dans le FIFO représente un client et ses items. On retient pour un client un vecteur de taille n, avec n le nombre d'items possibles dans le flux, et dans le vecteur on retient si le client a demandé cet item ou pas. Quand une transaction arrive (client c et item i), il y a deux possibilités :

1. Le client c existe dans le FIFO, alors on le déplace en tête (première place) du FIFO et on met à jour son vecteur d'items avec i.

2. Le client c n'existe pas dans le FIFO, alors on le crée en tête du FIFO et on ajoute i à son vecteur d'items.

Si la taille du FIFO dépasse sa taille max alors on supprime le dernier client (en queue de FIFO). L'exemple 10 illustre le fonctionnement du FIFO.

Exemple 10 *Considérons le flux composé des transactions $T1$ à $T6$ du tableau 6.1. A l'initialisation, le FIFO est composé d'un client ($C1$) avec un seul item (A). Puis, la transaction associant le client $C2$ et l'item A arrive dans le flux. Cette transaction est placée en début de FIFO. Le client $C1$ fait de nouveau une action avec l'item C et la transaction $(C1, \{A, C\})$ est placée en début de FIFO ($C1$ est déplacé de la queue vers la tête). Les mises à jour du FIFO continuent suivant ce principe.*

$T1$	$T2$	$T3$	$T4$	$T5$	$T6$
$C1$	$C2$	$C1$	$C3$	$C1$	$C2$
A	A	C	D	B	B

TAB. 6.1 – Un flux de transactions d'items.

$T1$

$C1$
A

$T2$

$C2$	$C1$
A	A

$T3$

$C1$	$C2$
A	A
C	

$T4$

$C3$	$C2$	$C1$
	A	A
	C	
D		

$T5$

$C1$	$C3$	$C2$
A		A
B		
C		
	D	

$T6$

$C2$	$C1$	$C3$
A	A	
B	B	
	C	
		D

TAB. 6.2 – Constitution du FIFO avec les transactions de la table 6.1.

Un objectif intéressant sur cette structure, serait de proposer une détection « au plus tôt ». Une approche naïve consisterait à utiliser un FIFO court, qui ne garderait donc que les événements les plus récents et permettrait d'extraire une connaissance à très court terme. Malheureusement, un

H

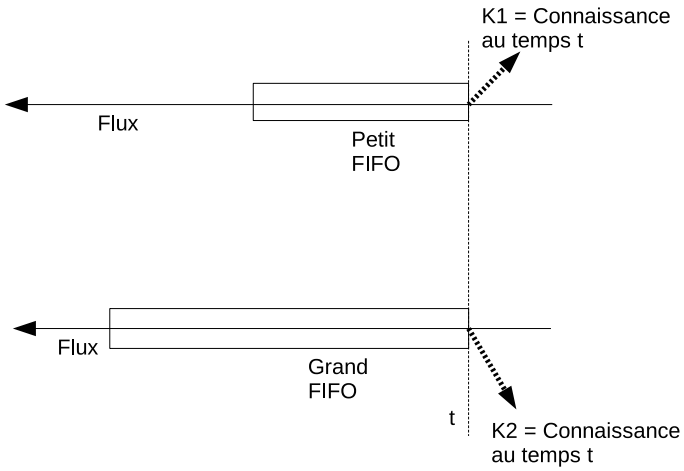

FIG. 6.1 – Impact de la taille de la fenêtre d'observation

FIFO court couperait la constitution des itemsets et aurait un impact négatif sur l'extraction.

Avec l'illustration de la figure 6.1, on peut facilement se convaincre qu'avec un petit FIFO, les itemsets stockés pour chaque client seront plus court et moins nombreux. Donc les itemsets extraits seront plus courts. Le nombre de motifs extraits sera également différent.

Il faudrait donc travailler avec un grand FIFO (le plus grand possible, *i.e.* « ce que permet la mémoire disponible »). D'un autre côté, un grand FIFO ne permet pas d'extraire facilement les itemsets fréquents (trop de données, trop de calcul).

Il s'agit donc de permettre cette extraction, sur des itemsets qui se constituent au cours du temps, tout en garantissant 1) des temps de calculs faibles et 2) d'obtenir cette extraction d'itemsets fréquents « au plus tôt ».

6.2.2 Extraction d'itemsets distinctifs

L'extraction d'itemsets distinctifs (ou informatifs) est un sujet de recherche récent qui connait plusieurs algorithmes pour les données statiques [KH06, HHM+07]. Ces solutions ne sont toutefois pas conçues pour le cas des flux de données, pour lesquels les temps de réponse doivent être aussi faibles

	A	B	C	D	E
D_1	1	0	1	1	1
D_2	1	1	0	0	1
D_3	1	0	0	1	1
D_4	1	0	1	1	1
D_5	1	1	0	1	1
D_6	0	1	1	1	1
D_7	0	0	1	1	1
D_8	0	1	0	1	1
D_9	0	0	1	1	1
D_{10}	0	1	0	1	1

FIG. 6.2 – Variables de documents

que possible. Dans cette piste de recherche (en cours d'étude dans la thèse de Chongsheng Zhang) nous considérons le problème de l'extraction d'itemsets distinctifs dans les flux, qui peut avoir de nombreuses applications dans la sélection de variables, la classification ou encore la recherche d'information. Ces itemsets sont différents des itemsets fréquents dans la mesure où ils minimisent la redondance tout en exprimant une information maximale sur les données. Cette information est mesurée en terme d'entropie.

Exemple 11 *Considérons une application de recherche de documents à partir des informations de la figure 6.2. Dans cette table, la valeur «1» signifie que la variable est présente dans le document et la valeur «0» signifie l'absence de cette variable. L'itemset (D,E) est un itemset fréquent car il apparait dans presque tous les documents. Toutefois, cet itemset n'est pas d'une grande aide pour la recherche d'un document (les variables ne permettent pas de distinguer les documents). En revanche, il est difficile de synthétiser les valeurs de l'itemset (A,B,C). Si on affecte à cet itemset la combinaison de valeurs (1,0,0) alors on peut trouver le document D_3 et avec les valeurs (0,1,1) on retrouve le document D_6. Bien que cet itemset soit non fréquent, étant donné que les items qui le composent ont souvent une combinaison de valeurs différentes d'un document à l'autre, il s'agit d'un itemset distinctif.*

L'exemple 11 illustre les applications possible en recherche d'information, mais ces applications couvrent de nombreux autres domaines, comme la selection de variables, la compression ou encore la classification. Nous

abordons le problème de l'extraction de ces motifs distinctifs dans les flux de données. Dans le cas des flux, le problème des techniques existantes, comme par exemple l'algorithme *ForwardSelection* proposé par [KH06] réside dans l'utilisation de plusieurs passes sur les données. Un des objectifs de la thèse Chongsheng Zhang est de proposer un algorithme fonctionnant en temps réel sur les flux pour l'extraction de ces itemsets distinctifs.

6.2.3 La temporalité dans une extraction en-ligne

Les travaux présentés dans la section 4.5 permettent l'extraction de motifs séquentiels (tenant compte de la temporalité) dans les flux avec un principe de découpage du flux par batches. Ce découpage par batches permet d'accumuler les transactions afin de constituer des séquences le temps d'une fenêtre. Une fois les séquences acquises, le clustering est appliqué. Malheureusement, dans ce cas de figure, c'est la taille de la fenêtre de temps qui décide de la façon dont les séquences se constituent.

La motivation de cette piste est similaire à la section 6.2.1 : éviter de couper des séquences en cours de constitution et ne pas considérer qu'elles arrivent de manière atomique.

Avec les travaux en cours de Chongsheng Zhang, nous voulons obtenir une extraction d'itemsets sur des transactions qui se constituent au fil du temps dans le flux. Ici, l'objectif serait d'obtenir une extraction de motifs séquentiels sur de telles transactions, alors que la section 6.2.1 présente des pistes pour extraire des itemsets sur ce type de structure. Dans ce cas, la difficulté associée viendrait non seulement de l'extraction des motifs, mais aussi de la gestion des données en entrée du flux (des séquences d'itemsets).

6.2.4 Différentes fonctions de vieillissement dans l'échantillonnage

Considérons à nouveau la structure de FIFO donnée en section 6.2.1. Nous avons vu qu'une extraction de motifs sur un FIFO de grande taille serait difficile à cause du temps de calcul incompatible avec les contraintes d'un flux. D'un autre côté, un petit FIFO ne permettrait pas d'avoir une fenêtre d'observation assez large pour obtenir des résultats pertinents.

L'échantillonnage peut répondre au problème du temps de calcul sur un FIFO de grande taille, mais il faudrait pousser la façon de le biaiser en fonc-

tion de la nature des résultats attendus. En effet, par sa nature le FIFO stocke en tête les événements les plus récents. En revanche, les événements sortants sont les plus « stables » (*i.e.* arrivés à maturité).

On peut alors envisager plusieurs façon de biaiser l'echantillonnage sur une telle structure :

- On peut considérer, par exemple, qu'un « accident » dans les motifs fréquent ne justifie pas de créer un nouveau modèle. Imaginons le cas des rafales (bursts) évoqué en section 4.3.2 et considérons le cas où ces événements ne doivent pas perturber le modèle. On veut alors simplement absorber ces événement dans le traitement. Pour cela on peut adopter une fonction de vieillissement très souple avec l'âge des données (on accepte plus d'événement anciens). Cela permettrait compenser les nouveaux arrivants qui génèrent beaucoup d'événements en tête de FIFO.
- Considérons maintenant le cas inverse, dans lequel on veut que tout changement soit répercuté immédiatement. Dans ce cas, la fonction sera très dure avec l'âge des données en utilisant une sélection privilégiant les récents. Par exemple une fonction fortement logarithmique pour les anciens. Ce type d'analyse peut trouver son utilité dans la surveillance de matériel ou dans des applications sécuritaires ou les changements de modèle doivent être repérés au plus vite.

6.2.5 Le problème des faux positifs

Les faux positifs sont une source d'erreur et de temps perdu dans de nombreux domaines. C'est le cas, par exemple, dans la publicité ciblée (pour laquelle je donne une illustration en section 6.2.5 où l'annonceur paie l'affichage de sa publicité avec des chances d'accrochage très très faibles). C'est également le cas dans la détection d'intrusion où des experts doivent analyser les résultats avant de prendre des décisions. Dans cette section j'expose deux pistes à explorer sur ce thème.

petite digression : voulez-vous acheter des briques ?

La *Sears Tower* a récemment fait l'actualité suite à une innovation architecturale. La célèbre tour de Chicago a effectivement été modifiée par l'ajout de deux balcons de verre, qui permettent de voir le sol comme si on était suspendu dans le vide à 412 mètres d'altitude (103^{eme} étage). En regardant les photos relatives à cette information, j'ai constaté que le site

hébergeant l'information proposait des publicité, via Google AdSense. La figure 6.3 montre une capture d'écran prise à cet instant. On y voit une photo de ces balcons avec des personnes qui profitent de cette vue particulière. On y voit également la publicité associée à cette photo. Cette publicité est certainement choisie en fonction de mots clés (ou « tags ») caractérisant la photo. Puisque les balcons sont construits à partir de briques de verre et que ces briques sont mises en valeur par les architectes du projet (sans elles, pas de balcon transparent) alors le mot « brique » est très important pour cette photo. C'est probablement la raison pour laquelle Google AdSense propose au lecteur d'aller chez « Point P » pour y acheter des briques...

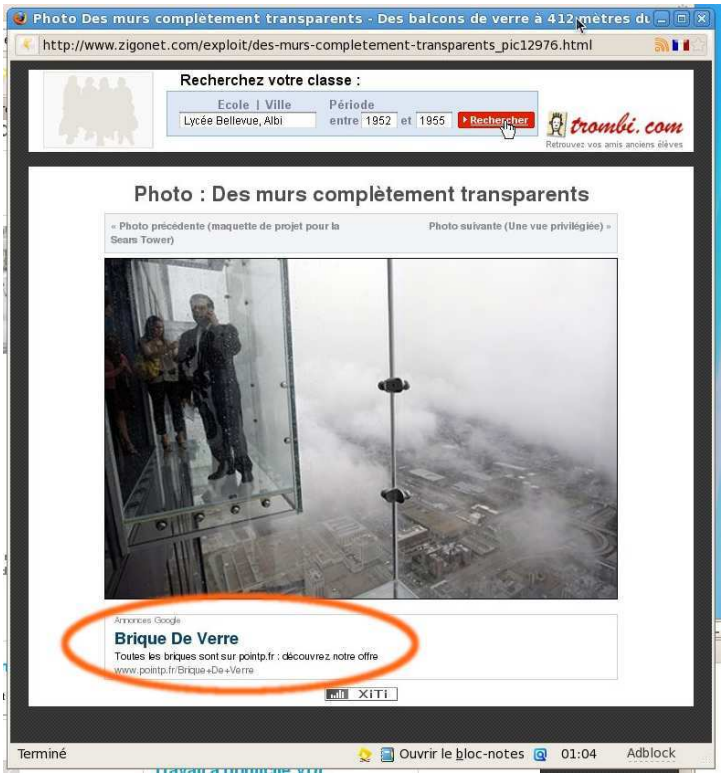

FIG. 6.3 – Vous pouvez faire la même chose... avec les bonnes briques !

Pistes envisagées

La nouveauté ET la contradiction.

Le principal problème des approches basées sur la détection d'anomalies dans une classification non-supervisée vient du nombre de faux positifs générés. Ces faux positifs s'expriment quand les anomalies sont exploitées en contexte réel (déclencher des alarmes, par exemple).

Un piste permettent de répondre à ce problème consisterait à mesurer la stabilité des anomalies (leur pourcentage par exemple). On peut alors décider que si les anomalies augmentent au delà d'un seuil fixé, le modèle doit être adapté.

Supposons alors qu'une ou plusieurs classes fassent leur apparition. Il serait possible d'accélerer les traitements en ne construisant que les classes qui contredisent le modèle précédent. Il s'agit alors d'éviter la reconstruction des classes connues avant d'appliquer la détection d'anomalies. L'objectif est alors d'intégrer les connaissances sur les classes du modèle précédent dans la génération des nouvelles classes.

Adapter la détection d'anomalies en fonction du contexte

Nous avons vu en section 5.4 l'utilité du contexte pour la détection d'intrusion (*i.e.* les anomalies partagées sur plusieurs sites sont quasi-systématiquement des intrusions). Il s'agit d'une première façon de baisser le taux de fausses alarmes en utilisant le contexte (les navigations et les requêtes sur un site Web). Les fausses alarmes ne sont pas un phénomène réservé aux seuls sites Web. La détection d'anomalies peut être employée dans d'autres contextes d'applications. Ainsi, il serait certainement pertinent d'ajouter des critères à la détection d'anomalies en utilisant des éléments venus des contextes produisant les données analysées. En effet, pour baisser le taux de faux positifs d'une détection d'anomalies basée sur la classification non-supervisée, les critères seront différents selon que l'analyse se fera sur des données financières, d'usages ou encore de la santé ou de l'environnement.

Chapitre 7

CV (septembre 2009)

Position actuelle

Chargé de recherche INRIA
Tel : 04 92 38 50 67
Fax : 04 92 38 77 55

Equipe AxIS
INRIA Sophia Antipolis
2004 route des lucioles – BP 93
06902 Sophia Antipolis

email : Florent.Masseglia@sophia.inria.fr
http ://www.inria.fr/sophia/teams/axis/personnel/Florent.Masseglia

Titres

- Depuis décembre 2006 : CR1 INRIA.
- De septembre 2002 à décembre 2006 : CR2 INRIA.

Equipe : AxIS (http ://www.inria.fr/sophia/teams/axis).

Diplômes

- Doctorat en Informatique obtenu le 07/01/2002 à L'université de Versailles St Quentin.
- D.E.A. d'informatique obtenu en 1998 à l'Université de Montpellier II.

Encadrement d'activités de recherche

Doctorants

- **Chongsheng Zhang** : Inria Sophia, depuis septembre 2008. Directeur de thèse sur autorisation de l'école doctorale STIC de Nice. Soutenance prévue en septembre 2011. Financement : ANR MIDAS (C.f. section « Actions nationales et internationales »). Sujet : « Résumé de flots de données par l'extraction de connaissances ».
- **Alice Marascu** : Inria Sophia, depuis septembre 2005. Participation à l'encadrement, sous la direction de Yves Lechevallier (DR Inria). Taux de participation à l'encadrement : 75%. Sujet : « Extraction de motifs séquentiels dans les flots de données ». Soutenue le 14 septembre 2009. J'ai dirigé les travaux d'Alice Marascu sur l'extraction de connaissances dans les flots de données et sur la gestion de l'historique de ces connaissances. Sur ces sujets, récents et majeurs dans la communauté, nous avons obtenu 15 publications à ce jour (2 revues internationales, 3 conférences internationales, 3 ateliers internationaux, 5 conférences nationales et 2 ateliers nationaux).
- **Doru Tanasa** : Inria Sophia, janvier 2003 à juin 2005. Participation à l'encadrement, sous la direction de Brigitte Trousse. J'ai participé (15%) à l'encadrement de la thèse de Doru Tanasa sur les aspects « méthodes hybrides d'extraction de connaissances ». Nos travaux sur ce sujet ont donné lieu à 5 publications (1 conférence internationale, 1 chapitre de livre international et 3 conférences nationales).

Post-doctorants

- **Céline Fiot**. Post-Doctorante dans l'équipe AxIS (financement Inria). De Septembre 2008 à Janvier 2009. Sujet : fouille de données graduelle pour la détection d'intrusions. Co-encadrée avec M. Teisseire et A. Laurent du Lirmm. Taux d'encadrement : 50%. J'ai dirigé les travaux de Céline Fiot sur les aspects sécurité et motifs séquentiels. Nos travaux ont donné lieu à 4 publications (1 revue internationale, 2 conférences internationales majeures dans le domaine du graduel : FuzzIEEE et IPMU, et 1 conférence nationale).
- **Wei Wang**. De janvier 2008 à décembre 2008. Post-Doctorant dans l'équipe AxIS, projet « SéSur » (financement ARC Inria). Sujet : Fouille de flots de données pour la détection d'intrusions. Co-encadré avec M.O Cordier, R. Quiniou et B. Trousse. Taux d'encadrement : 25%. Nos travaux ont donné lieu à 1 publication nationale (EGC'09).

Etudiants de Master

- Bashar Saleh, UNSA, 2007.Taux d'encadrement : 100%.
- Sofiane Sellah, Master ECD, Univ. Lyon 2. 2005.
  Taux d'encadrement : 50% (avec Brigitte Trousse).
- Alice Marascu, UNSA. 2005. Taux d'encadrement : 100%.
- Calin Garboni, Université de Timisoara. 2005.
  Taux d'encadrement : 50% (avec Brigitte Trousse).
- Calin Garboni, UNSA. 2004. Taux d'encadrement : 100%.
- Pierre Alain Laur : Université de Montpellier II, 2002,
  Taux d'encadrement 20% (avec Pascal Poncelet).
- Simon Jaillet, Université de Montpellier II. 2001.
  Taux d'encadrement : 50% (avec Pascal Poncelet).

Elèves ingénieurs

- 1 ingénieur troisième année, Fabien Benoit, I3S Sophia Antipolis. 2003.
  Taux d'encadrement : 100%.
- 3 ingénieurs deuxième année, en 2001 et 2003, ISI Montpellier. Taux d'encadrement : 50% (avec Pascal Poncelet).

Internships (stages d'Universités/Ecoles étrangères)

Les stagiaires suivants sont co-encadrés avec Pascal Poncelet (LIRMM), dans le cadre du projet Color MUTAN.

- Goverdhan Singh : IIT Jaipur (Inde), mai-juillet 2008. Equipe AxIS, INRIA Sophia Antipolis.
- Gaurav Panthari : IIT Jaipur (Inde), mai-juillet 2008. Equipe AxIS, INRIA Sophia Antipolis.
- Nischal Verma : IIT Guwahati (Inde), mai-juillet 2008. Equipe KDD, LGI2P Nîmes.
- Dipankar Das : IIT Guwahati (Inde), mai-juillet 2008. Equipe KDD, LGI2P Nîmes.

Enseignements

- Data Mining (Niveau Master, 23 heures) : UNSA, de 2004 à 2008 et Univ. Montpellier II de 1999 à 2001.
- Systèmes d'exploitation (IUP 2ème Année, 104 heures) : TD pour L'Univ. Montpellier II. 1998 et 1999.

- C++ (IUT et Maîtrise, 64 heures) : Univ. Aix en Pce en 1998 et Univ. Montpellier II en 2001.
- Bases de Données (2ème année ingénieurs, 130 heures) : ISIM Montpellier en 1999, 2000 et 2001.
- Informatique pour tous (1ère année Deug, 108 heures) : Univ. Montpellier III. 1999 et 2000

Actions nationales et internationales

Projets de recherche avec financement

- **Projet ANR MIDAS** « Mining Data Streams ». Participant à 40%, avec un financement destiné à accueillir un doctorant sur la période 2008-2011. Responsable scientifique pour l'Inria sur ce projet. Membre du comité de gestion et du comité de pilotage.
- **ARC Inria SéSur** « Sécurité et Surveillance dans les flots de données ». Porteur du projet avec financements pour un post-doc et 4 stagiaires de Master sur la période 2007-2008.
- **Color Inria MUTAN** « Mutualisation des Anomalies pour la détection d'intrusions ». Porteur du projet avec financements pour 4 internships sur l'année 2008.

Animation de la communauté

J'ai co-présidé les événements scientifiques suivants :

- 7ème atelier international MDM (Multimedia Data Mining) en conjonction avec la conférence KDD'06, Philadelphie. Avec Zhongfei Zhang, Ramesh Jain et Alberto Del Bimbo.
- 6ème atelier international MDM (Multimedia Data Mining) en conjonction avec la conférence KDD'05, Chicago. Avec Latifur Kahn et Fatma Bouali.
- 4ème atelier national FDC (Fouille de Données Complexes) en conjonction avec la conférence EGC'07, Namur. Avec Omar Boussaid.
- 2ème atelier national FDC (Fouille de Données Complexes) en conjonction avec la conférence EGC'05, Paris. Avec Pierre Gançarski.

J'ai participé aux comités d'organisation des conférences EGC'02 et EGC'08.

J'ai été Co-animateur (avec O. Boussaïd, Univ. Lyon 2) du thème de travail « Structuration et Organisation des Données Complexes » dans le

cadre du Groupe de Travail « Fouille de Données Complexes » (de 2003 à 2008).

Activité d'édition

- Ouvrage international chez IGI Publisher sur le thème « Data Mining Patterns : New Methods and Applications ». En cours de finalisation. Parution prévue en 2007. Avec P. Poncelet et M. Teisseire (LIRMM, Montpellier).
- Ouvrage international chez IGI Publisher sur le thème « Successes and New Directions in Data Mining ». En cours de finalisations. Parution prévue en 2007. Avec P. Poncelet et M. Teisseire (LIRMM, Montpellier).
- Numéro spécial de la revue internationale IEEE Transactions on Multimedia (T-MM) sur le thème « Multimedia Data Mining ». Volume 10, Number 2, February 2008. Avec Zhongfei (Mark) Zhang, Ramesh Jain et Alberto Del Bimbo.
- Numéro spécial de la revue internationale Multimedia Tools and Applications (MTAP) sur le thème « Multimedia Data Mining ». Volume 35 :1. October 2007. Avec Latifur Kahn et Fatma Bouali.
- Numéro spécial de la revue nationale RNTI (Cépaduès) sur la fouille de données complexes. Avec O. Boussaid (Univ. Lyon 2), P. Gançarski (Univ. Strasbourg) et B. Trousse (Inria Sophia). 2005

Comités de programme

- Conférences internationales :
  - 2009 : ACM SAC (Track on Data Streams), Discovery Science, ISICA
  - 2008 : ICDM, MCCSIS (IADIS), e-Commerce
  - 2007 : ICDM, e-Commerce
  - 2006 : ICTAI'06
- Conférences nationales :
  - 2009 : EGC
  - 2008 : EGC, BDA
  - 2006 : BDA
  - 2004 : BDA
  - 2003 : Inforsid
- Ateliers internationaux : PIKM'08, PIKM'09, MSTD'05, ILO'07.

– Présidences de session : Time 08, BDA 06, Inforsid 05.

Relectures et expertises

Relecteur pour :
– 12 journaux internationaux : DMKD, KAIS, TKDE, DKE, IS, INS, ETRI, CI, JSS, JIIS, JAIR, ESWA.
– 5 ouvrages internationaux : « E-Strategies for Resource Management Systems » (IGI, prévu pour 2010), « Advances in Knowledge Discovery and Management », (Springer, 2009), « Encyclopedia of Multimedia Technology and Networking » (IGI, 2006), « Encyclopedia of Data Warehousing and Mining » (IGI, 2004, 2006) et « Processing and Managing Complex Data for Decision Support » (IGI, 2005).
– 1 revue nationale : Second numéro special RNTI sur la fouille de données complexes (Ed. Boussaid, Gançarski et B. Trousse). 2008.

Expertises :
– Expert pour l'ANR, appel CONTINT 2009.
– Expert pour l'attribution d'une chaire internationale, région Nord-Pas de Calais, 2007.

Liste de publications

Les publications indiquées par un ✓ sont issues de travaux conduits après le doctorat (*i.e.* les travaux que j'ai dirigés ou co-dirigés à l'Inria). Les publications indiquées par un ✗ sont issues de mes travaux de doctorat.

Edition d'ouvrages nationaux et internationaux

- ✓ Zhongfei Zhang, Florent Masseglia, Ramesh Jain, Alberto Del Bimbo. Special Issue on Multimedia Data Mining. IEEE Transactions on Multimedia (TMM) Journal 10(2) : 165-166. 2008
- ✓ F. Bouali, F. Masseglia, L. Khan (editors) Special Issue on Multimedia Data Mining (MDM/KDD'06), Multimedia Tools and Applications (MTAP), Springer, vol. 35 :1, 2007.
- ✓ O. Boussaid, F. Masseglia (editors) Actes de FDC'07, le quatrième atelier sur la « Fouille de données complexes dans un processus d'extraction de connaissances », 2007.
- ✓ F. Masseglia, P. Poncelet, M. Teisseire (editors) Successes and New Directions in Data Mining, Information Science Reference, ISBN 978-1599046457, Idea Group, 2007.
- ✓ P. Poncelet, F. Masseglia, M. Teisseire (editors) Data Mining Patterns : New Methods and Applications, Premier Reference Source, ISBN 978-1599041629, Idea Group, 2007.
- ✓ Zhongfei Zhang, Florent Masseglia, Ramesh Jain and Alberto Del Bimbo. Proceedings of MDM'06, the seventh international workshop on « Multimedia Data Mining » (held in conjunction with KDD'06). Philadelphia, USA, August 2006.
- ✓ O. Boussaïd, P. Gançarski, F. Masseglia and B. Trousse (rédacteurs invités). Revue des Nouvelles Technologies de l'Information (RNTI). Numéro spécial « Fouille de données complexes ». Cépaduès éditions. Vol 7. 2005.
- ✓ F. Bouali, L. Kahn and F. Masseglia. Proceedings of MDM'05, the sixth international workshop on « Multimedia Data Mining » (held in conjunction with KDD'05). Chicago, USA, August 2005.
- ✓ P. Gançarski et F. Masseglia. Actes de FDC'05, le second atelier sur la « Fouille de données complexes dans un processus d'extraction de connaissances ». (Atelier de la conférence EGC'05). Paris. Janvier 2005.

Articles dans des revues internationales avec comité de lecture

- ✓ Alice Marascu and Florent Masseglia. « Atypicity Detection in Data Streams : a Self-Adjusting Approach ». In Intelligent Data Analysis Journal (IDA). To appear. 2009.
- ✓ Céline Fiot, Florent Masseglia, Anne Laurent and Maguelonne Teisseire. « Evolution Patterns and Gradual Trends ». In International Journal of Intelligent Systems (IJIS). 2009.
- ✓ F. Masseglia, P. Poncelet, and M. Teisseire, « Efficient mining of sequential patterns with time constraints : Reducing the combinations ». In Expert Systems with Applications (ESWA) Journal. Elsevier. Vol 36, Issue 2. March 2009.
- ✓ Florent Masseglia and Pascal Poncelet and Maguelonne Teisseire and Alice Marascu. « Web Usage Mining : Extracting Unexpected Periods from Web Logs ». In Data Mining and Knowledge Discovery (DMKD) Journal. Springer Netherlands (Ed). 16(1) : 39-65 (2008).
- ✓ Alice Marascu and Florent Masseglia. « Mining Sequential Patterns from Data Streams : a Centroid Approach ». In Journal for Intelligent Information Systems (JIIS). Issue 27, Number 3, pp 291-307. November 2006.
- ✗ F. Masseglia and M. Teisseire and P. Poncelet. « HDM : A client/server/engine architecture for real time web usage mining ». In Knowledge and Information Systems (KAIS), Volume 5, Number 4, pp 439 - 465, November 2003.
- ✗ F. Masseglia and P. Poncelet, and M. Teisseire « Incremental mining of sequential patterns in large databases ». In Data & Knowledge Engineering (DKE), Vol. 46 (2003), pp. 97-121, July 2003.
- ✗ F. Masseglia, P. Poncelet, and R. Cicchetti. « An Efficient Algorithm for Web Usage Mining ». In Networking and Information Systems (NIS), Vol. 2, N. 5-6, pp. 571-603, December 1999.

Articles invités dans des revues internationales

- ✓ Zhongfei Zhang, Florent Masseglia, Ramesh Jain and Alberto Del Bimbo. « Report on MDM/KDD2006 : The seventh International Workshop on Multimedia Data Mining ». SIGKDD Explorations 8(2) : 92-95. 2006.
- ✓ F. Bouali, L. Kahn and F. Masseglia. Report on MDM/KDD2005 :

The sixth International Workshop on Multimedia Data Mining. SIGKDD Explorations 7(2) : 148-150. 2005.

✗ F. Masseglia, P. Poncelet, and M. Teisseire. « Using Data Mining Techniques on Web Access Logs to Dynamically Improve Hypertext Structure ». In ACM SigWeb Letters , pp. 13-19, Vol. 8, N. 3, October 1999.

Chapitres de livres d'audience internationale

✓ Goverdhan Singh, Florent Masseglia, Céline Fiot, Alice Marascu and Pascal Poncelet. « Mining Common Outliers for Intrusion Detection ». Advances in Knowledge Discovery and Management (AKDM09), post-actes EGC'09. Accepté sous réserve de modifications. A paraître.

✓ Nischal Verma, François Trousset, Pascal Poncelet and Florent Masseglia. « Intrusion Detections in Collaborative Organizations by Preserving Privacy ». Advances in Knowledge Discovery and Management (AKDM09), post-actes EGC'09. A paraître.

✓ Doru Tanasa, Florent Masseglia and Brigitte Trousse. « Mining Generalized Web Data for Discovering Usage Patterns ». In Encyclopedia of Data Warehousing and Mining - 2nd Edition. Information Science Publishing. ISBN : 978-1-60566-010-3. August 2008.

✓ B. Trousse, M.-A. Aufaure, B. Le Grand, Y. Lechevallier, F. Masseglia. « Web Usage Mining for Ontology Management », in : Data Mining with Ontologies : Implementations, Findings and Frameworks. N. Hèctor Oscar, Gonzalez Cisaro. Sandra Elisabeth G, X. Daniel Hugo (editors), Information Science Reference, 2007, chap. 3, p. 37-64.

✓ Florent Masseglia, Pascal Poncelet and Maguelonne Teisseire. « Peer-to-Peer Usage Analysis ». In Encyclopedia of Multimedia Technology and Networking, M. Pagani (Ed.), 16 pages, IRM Press, Hershey PA, 2005.

✓ F. Masseglia, M. Teisseire, and P. Poncelet. « Sequential Pattern Mining : A Survey on Issues and Approaches ». Encyclopedia of Data Warehousing and Mining. Information Science Publishing. ISBN : 1-59140-557-2. 2005

Articles dans des conférences internationales avec comité de sélection

- ✓ Wang, W., Masseglia, F., Guyet, T., Quiniou, R., and Cordier, M. 2009. A general framework for adaptive and online detection of web attacks. In Proceedings of the 18th international Conference on World Wide Web (WWW '09, poster). Madrid, Spain, April 20 - 24, 2009.
- ✓ Alice Marascu and Florent Masseglia. « A Multi-Resolution Approach for Atypical Behaviour Mining ». In 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'09). Bangkok, Thailand, April 2009.
- ✓ G. Singh, F. Masseglia, C. Fiot, A. Marascu and P. Poncelet. « Data Mining for Intrusion Detection : from Outliers to True Intrusions. » In 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'09). Bangkok, Thailand, April 2009.
- ✓ Alice Marascu and Florent Masseglia. « Parameterless Outlier Detection in Data Streams ». In 24th Annual ACM Symposium on Applied Computing (ACM SAC'09). Honolulu, Hawaii, USA, March 2009.
- ✓ C. Fiot, F. Masseglia, A. Laurent, and M. Teisseire. Gradual trends in fuzzy sequential patterns. 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'08). Malaga, Spain, June 2008.
- ✓ C. Fiot, F. Masseglia, A. Laurent, and M. Teisseire. TED and EVA : Expressing Temporal Tendencies Among Quantitative Variables Using Fuzzy Sequential Patterns. 17th IEEE International Conference on Fuzzy Systems (FuzzIEEE'08), Hong Kong, June 2008.
- ✓ B. Saleh and F. Masseglia. « Time Aware Mining of Itemsets ». In Proceedings of the Fifteenth International Symposium on Temporal Representation and Reasoning (TIME 08), Montréal, Jun 16-18. 2008.
- ✓ Florent Masseglia, Pascal Poncelet and Maguelonne Teisseire. « Peer-to-Peer Usage Analysis : a Distributed Mining Approach ». 20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06), pp - 993-998, Vienna, April, 2006.
- ✓ Florent Masseglia, Maguelonne Teisseire et Pascal Poncelet. « Pre-Processing Time Constraints for Efficiently Mining Generalized Sequential Patterns ». In 11th International Symposium on Temporal Representation and Reasoning (TIME'04), Tatihou, France, 1-3 July, 2004.

✓ F. Masseglia, D. Tanasa, and B. Trousse. « Web Usage Mining : Sequential Pattern Extraction with a Very Low Support ». In Advanced Web Technologies and Applications : 6th Asia-Pacific Web Conference , APWeb 2004, Hangzhou, China, April 14-17, 2004. Proceedings, volume 3007 of LNCS, pages 513-522, 2004. Springer-Verlag.

✗ F. Masseglia and M. Teisseire and P. Poncelet. « Real Time Web Usage Mining with a Distributed Navigation Analysis » In Proceedings of the 12th International Workshop on Research Issues on Data Engineering (RIDE'02), San Jose, USA, February 2002.

✗ F. Masseglia and M. Teisseire and P. Poncelet. « Real Time Web Usage Mining : a Heuristic based Distributed Miner ». In Proceedings of Web Information Systems Engineering (WISE'01), Kyoto, Japan, December 2001.

✗ P.A. Laur, F. Masseglia, P. Poncelet, and M. Teisseire. « A General Architecture for Finding Structural Regularities on the Web ». Proceedings of the 9th International Conference on Artificial Intelligence (AIMSA'00), Lecture Notes in Artificial Intelligence, Springer Verlag, Varna, Bulgaria, September 2000.

✗ P.A. Laur, F. Masseglia, and P. Poncelet. « Schema Mining : Finding Structural Regularity among Semi structured Data ». Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'2000), Poster session, Lecture Notes in Artificial Intelligence, Springer Verlag, Lyon, France, September 2000.

✗ F. Masseglia, P. Poncelet, and M. Teisseire. « Web Usage Mining : How to Efficiently Manage New Transactions and New Clients ». Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'2000), Poster session, Lecture Notes in Artificial Intelligence, Springer Verlag, Lyon, France, September 2000.

✗ F. Masseglia, P. Poncelet, and R. Cicchetti. « WebTool : An Integrated Framework for Data Mining ». Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA'99), Lecture Notes in Computer Science, Springer Verlag, Vol. 1677, pp. 892-901, Florence, Italy, September 1999.

✗ F. Masseglia, F. Cathala, and P. Poncelet. « The PSP approach for mining sequential patterns ». Proceedings of the 2nd European Conference on Principles of Data Mining and Knowledge Discovery in Databases (PKDD'98), Lecture Notes in Artificial Intelligence, Springer

Verlag, pp. 176-184, Nantes, France, September 1998.

Articles dans des conférences nationales avec comité de sélection

- ✓ Nischal Verma, François Trousset, Pascal Poncelet, Florent Masseglia : « Détection d'intrusions dans un environnement collaboratif sécurisé ». In Extraction et gestion des connaissances (EGC 2009), Strasbourg, January 2009. pp 301-312. (Nominé pour le prix du meilleur papier applicatif).
- ✓ Goverdhan Singh, Florent Masseglia, Celine Fiot, Alice Marascu, Pascal Poncelet : « Collaborative Outlier Mining for Intrusion Detection ». In Extraction et gestion des connaissances (EGC 2009), Strasbourg, January 2009. pp 313-324. (Nominé pour le prix du meilleur papier applicatif).
- ✓ Alice Marascu, Florent Masseglia : « Détection d'enregistrements atypiques dans un flot de données : une approche multi-résolution ». In Extraction et gestion des connaissances (EGC 2009), Strasbourg, January 2009. pp 455-456.
- ✓ Wei Wang, Thomas Guyet, Rene Quiniou, Marie-Odile Cordier, Florent Masseglia : « Online and adaptive anomaly Detection : detecting intrusions in unlabelled audit data streams ». In Extraction et gestion des connaissances (EGC 2009), Strasbourg, January 2009. pp 457-458.
- ✓ B. Saleh and F. Masseglia. Extraction de motifs séquentiels compacts. Extraction et Gestion des Connaissances (EGC 08), Sophia-Antipolis, Jan. 29-Feb 1st 2008.
- ✓ C. Fiot, F. Masseglia, A. Laurent and M. Teisseire. Séquences, tendances et connaissances. 26ème Congrès Informatique des organisations et systèmes d'information et de décision (Inforsid'08), 2008.
- ✓ Doru Tanasa, Florent Masseglia, Brigitte Trousse. « GWUM : une généralisation des pages Web guidée par les usages », INFORSID Informatique des organisations et systèmes d'information et de décision, Hammamet, Tunisie, June 2006.
- ✓ Alice Marascu, Florent Masseglia. « Classification de flots de séquences basée sur une approche centroïde », INFORSID Informatique des organisations et systèmes d'information et de décision, Hammamet, Tunisie, June 2006.
- ✓ Alice Marascu, Florent Masseglia. « Extraction de motifs séquentiels

dans les flots de données d'usage du Web », Extraction et Gestion des Connaissances (EGC'06), Lille, January 2006.

✓ Florent Masseglia, Pascal Poncelet, Maguelonne Teisseire and Alice Marascu. « Usage Mining : extraction de périodes denses à partir des logs Web ». Extraction et Gestion des Connaissances (EGC'06), Lille, January 2006.

✓ Florent Masseglia, Pascal Poncelet, Maguelonne Teisseire. « Fouille de données dans les systèmes Pair-à-Pair pour améliorer la recherche de ressources », Extraction et Gestion des Connaissances (EGC'06), Lille, January 2006.

✓ D. Tanasa, B. Trousse et Florent Masseglia « Classer pour Découvrir : une nouvelle méthode d'analyse du comportement de tous les utilisateurs d'un site Web ». In Extraction et Gestion des Connaissances EGC'2004, pp 549-560, January 2004.

✓ F. Masseglia and D. Tanasa, and B. Trousse « Diviser pour Découvrir : une Méthode d'Analyse du Comportement de Tous les Utilisateurs d'un Site » In Actes des 19ièmes Journées Bases de Données Avancées (BDA'03), Lyon, France, Octobre 2003.

✗ F. Masseglia and M. Teisseire and P. Poncelet « Web Usage Mining Intersites : Analyse du Comportement des Utilisateurs à Impact Immédiat ». Actes des 17ièmes Journées Bases de Données Avancées (BDA'01), Agadir, Maroc, Octobre 2001

✗ F. Masseglia, P. Poncelet and M. Teisseire. « Incremental Mining of Sequential Patterns in Large Databases ». Actes des 16ièmes Journées Bases de Données Avancées (BDA'00), Blois, France, Octobre 2000.

✗ F. Masseglia, P. Poncelet et M. Teisseire. « Extraction efficace de motifs séquentiels : le prétraitement des données ». Actes des 15ièmes Journées Bases de Données Avancées (BDA'99), pp. 341-360, Bordeaux, France, Octobre 1999.

✗ F. Masseglia, P. Poncelet et R. Cicchetti. « Analyse du comportement des utilisateurs sur le Web ». Actes du 17ième Congrés Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID'99), Toulon, France, Juin 1999.

Articles dans des ateliers internationaux avec comité de sélection

- ✓ Calin Garboni, Florent Masseglia and Brigitte Trousse. « Sequential Pattern Mining for Structure-Based XML Document Classification ». In 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, (INEX) 2005, Dagstuhl Castle, Germany, November 28-30, 2005, Revised Selected Papers. Lecture Notes in Computer Science 3977 Springer 2006, ISBN 3-540-34962-6.
- ✓ F. Masseglia, P. Poncelet, M. Teisseire and A. Marascu. « Web Usage Mining : Extracting Unexpected Periods from Web Logs ». In IEEE 2nd Workshop on Temporal Data Mining (TDM'05). Held in conjunction with ICDM'05, Houston, USA, November 27, 2005.
- ✓ A. Marascu and F. Masseglia. « Mining Data Streams for Frequent Sequences Extraction ». In IEEE first Workshop on Mining Complex Data (MCD'05). Held in conjunction with ICDM'05, Houston, USA, November 27, 2005.
- ✓ A. Marascu and F. Masseglia. « Mining Sequential Patterns from Temporal Streaming Data ». In ECML/PKDD first Workshop on Mining Spatio-Temporal Data (MSTD'05). Held in conjunction with PKDD'05, Porto, Portugal, October 3-7, 2005.
- ✓ C. Garboni and F. Masseglia. « Structure Mining - A Sequential Pattern based Approach ». In 6th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2004, Timisoara, Romania, 26-30 September 2004.

Article dans un atelier national

- ✓ Marascu, F. Masseglia. « Limites d'une approche incrémentale pour la segmentation de séquences dans les flux ». In : Atelier Flux de Données, Namur, Belgique, 23 January 2007, p. 49-60.

Bibliographie

[AFR97] E. Aleskerov, B.and Freisleben, and B. Rao. Cardwatch : A neural network based database mining system for credit card fraud detection. In *IEEE Computational Intelligence for Financial Engineering*, 1997.

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD Conf.*, pages 207–216, Washington DC, USA, May 1993.

[AM04] Arvind Arasu and Gurmeet Singh Manku. Approximate counts and quantiles over sliding windows. In *PODS '04 : Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 286–296, New York, NY, USA, 2004. ACM.

[AR00] Juan M. Ale and Gustavo H. Rossi. An approach to discovering temporal association rules. pages 294–300, 2000.

[AS95] R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proceedings of the 11th Int. Conf. on Data Engineering (ICDE'95)*, Tapei, Taiwan, March 1995.

[AY01] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. *SIGMOD Records*, 30(2) :37–46, 2001.

[BCH+01] E. Bloedorn, A. D. Christiansen, W. Hill, C. Skorupka, and L. M. Talbot. Data mining for network intrusion detection : How to get started. Technical report, MITRE, 2001.

[BDMO03] Brain Babcock, Mayur Datar, Rajeev Motwani, and Liadan O'Callaghan. Maintaining variance and k-medians over data stream windows. In *PODS '03 : Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 234–243, New York, NY, USA, 2003. ACM.

[BGG+01] F. Bonchi, F. Giannotti, C. Gozzi, G. Manco, M. Nanni, D. Pedreschi, C. Renso, and S. Ruggieri. Web log data warehousing and mining for intelligent web caching. *Data Knowledge Engineering*, 39(2) :165–189, 2001.

[BHV00] N. Billor, A. S. Hadi, and P. F. Velleman. BACON : blocked adaptive computationally efficient outlier nominators. *Computational Statistics and Data Analysis*, 34, 2000.

[BKNS00] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof : identifying density-based local outliers. *SIGMOD Records*, 29(2) :93–104, 2000.

[BT02] A. Benedek and B. Trousse. Adaptation of Self-Organizing Maps for CBR case indexing. In *Proceeding of the 4th International Workshopon Symbolic and Numeric Algorithms for Scientific Computing*, pages 31–45, Timisoara, Romania, October 2002.

[BWJ01] D. Barbara, N. Wu, and S. Jajodia. Detecting novel network intrusions using bayes estimators. In *1st SIAM Conference on Data Mining*, 2001.

[CAMSC05] W. Chimphlee, A. H. Abdullah, M. N. Md Sap, and S. Chimphlee. Unsupervised anomaly detection with unlabeled data using clustering. In *International conference on information and communication technology*, 2005.

[CBT04] Sergiu Chelcea, Patrice Bertrand, and Brigitte Trousse. A new agglomerative 2-3 hierarchical clustering algorithm. In *Innovations in Classification, Data Science, and Information Systems. Proc. 27th Annual GfKl Conference, University of Cottbus, March 12-14, 2003*, pages 3–10, Heidelberg-Berlin, Germany, 2004. Springer-Verlag.

[CDG07] Toon Calders, Nele Dexters, and Bart Goethals. Mining frequent itemsets in a stream. In *ICDM*, pages 83–92, 2007.

[CDH+na] Y. Chen, G. Dong, J. Han, B. Wah, and J. Wang. Multi-dimensional regression analysis of time-series data streams, VLDB 2002. Hong-Kong, China.

[CFL05] Philip S. Yu Cheong Fung, Jeffrey Xu Yu and Hongjun Lu. Parameter free bursty events detection in text streams. In *VLDB '05 : Proceedings of the 31st international conference on Very large data bases*, pages 181–192, 2005.

[Che07] Sergiu Chelcea. *Agglomerative 2-3 Hierarchical Classifications : Theoretical and Applicative Study*. PhD thesis, Universite de Nice Sophia Antipolis - France, 2007.

[CHM+00] Igor V. Cadez, David Heckerman, Christopher Meek, Padhraic Smyth, and Steven White. Visualization of navigation patterns on a web site using model-based clustering. In *Proceedings of the sixth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 280–284, Boston, Massachusetts, 2000.

[CL03] Joong Hyuk Chang and Won Suk Lee. Finding recent frequent itemsets adaptively over online data streams. In *KDD '03 : Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 487–492, 2003.

[CL04] Joong Hyuk Chang and Won Suk Lee. A sliding window method for finding recently frequent itemsets over online data streams. *J. Inf. Sci. Eng.*, 20(4) :753–762, 2004.

[CL05] Joong Hyuk Chang and Won Suk Lee. estwin : Online data stream mining of recent frequent itemsets by sliding window method. *J. Inf. Sci.*, 31(2) :76–90, 2005.

[CLR] T. Cormen, C. Leiserson, and R. Rivest. *Introduction a l'Algorithmique*. ed. Dunod.

[CMS99] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1) :5–32, 1999.

[Con98] World Wide Web Consortium. httpd-log files. In *http ://lists.w3.org/Archives*, 1998.

[CP99] Xiaodong Chen and Ilias Petrounias. Mining temporal features in association rules. In *PKDD '99 : Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, pages 295–300, 1999.

[CS03] Edith Cohen and Martin Strauss. Maintaining time-decaying stream aggregates. In *PODS '03 : Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 223–233, San Diego, California, 2003.

[CWYM04] Yun Chi, Haixun Wang, Philip S. Yu, and Richard R. Muntz. Moment : Maintaining closed frequent itemsets over a stream sliding window. *icdm*, 00 :59–66, 2004.

[CYL+05] Zhihong Chong, Jeffrey Xu Yu, Hongjun Lu, Zhengjie Zhang, and Aoying Zhou. False-Negative Frequent Items Mining from Data Streams with Bursting. In *DASFAA'05 : Database Systems for Advanced Applications*, pages 422–434, 2005.

[Dau92] Ingrid Daubechies. *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[DEK+02] P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava, and P. Tan. Data mining for network intrusion detection. In *NSF Workshop on Next Generation Data Mining*, 2002.

[EAP+02] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S Stolfo. A geometric framework for unsupervised anomaly detection : Detecting intrusions in unlabeled data. *Applications of Data Mining in Computer Security*, 2002.

[EEL+04] L. Ertoz, E. Eilertson, A. Lazarevic, P.-N. Tan, V. Kumar, J. Srivastava, and P. Dokas. Minds - minnesota intrusion detection system. *Data Mining - Next Generation Challenges and Future Directions*, 2004.

[FMLT08] Céline Fiot, Florent Masseglia, Anne Laurent, and Maguelonne Teisseire. Des séquences aux tendances. In *INFORSID*, 2008.

[FPSSU96] U.M. Fayad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA, 1996.

[FSS00] Y. Fu, K. Sandhu, and M. Shih. A generalization-based approach to clustering of web usage sessions. In *Proceedings of the 1999 KDD Workshop on Web Mining, San Diego, CA. Springer-Verlag*, volume 1836 of *LNAI*, pages 21–38. Springer, 2000.

[FYM05] R. Fujimaki, T. Yairi, and K. Machida. An approach to spacecraft anomaly detection problem using kernel feature space. In *11th ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005.

[FZFW06] H. Fan, O. R. Zaiane, A. Foss, and J. Wu. A nonparametric outlier detection for effectively discovering top-n outliers from engineering data. In *Pacific-Asia conference on knowledge discovery and data mining*, 2006.

[GGR02] Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. 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*, 2002.

[GHP+03] C. Giannella, J. Han, J. Pei, X. Yan, and P.S. Yu. *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, 2003.

[hA] http Analyze. http ://www.http-analyze.org/.

[HHM+07] Hannes Heikinheimo, Eino Hinkkanen, Heikki Mannila, Taneli Mielikäinen, and Jouni K. Seppänen. Finding low-entropy sets and trees from binary data. In *KDD '07 : Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 350–359, New York, NY, USA, 2007. ACM.

[HLS] SY Lee HF Li and MK Shan. Dsm-fi : An efficient algorithm for mining frequent itemsets in data streams. Accepted and to appear in Knowledge and Information Systems (KAIS).

[HWV02] Birgit Hay, Geert Wets, and Koen Vanhoof. Web Usage Mining by Means of Multidimensional Sequence Alignment Method. In *WEBKDD*, pages 50–65, 2002.

[HWV04] B. Hay, G. Wets, and K. Vanhoof. Mining Navigation Patterns Using a Sequence Alignment Method. *Knowl. Inf. Syst.*, 6(2) :150–163, 2004.

[HXD03] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24, 2003.

[JA05] Ruoming Jin and Gagan Agrawal. An algorithm for in-core frequent itemset mining on streaming data. *icdm*, 0 :210–217, 2005.

[JCLR07] Long Jin, Duck Jin Chai, Yang Koo Lee, and Keun Ho Ryu. Mining frequent itemsets over data streams with multiple time-sensitive sliding windows. *Advanced Language*

Processing and Web Information Technology, 2007. ALPIT 2007. Sixth International Conference on, pages 486–491, 22-24 Aug. 2007.

[JORL04] J. Joshua Oldmeadow, S. Ravinutala, and C. Leckie. Adaptive clustering for network intrusion detection. In *8th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, volume 3056 of *Lecture Notes in Computer Science*, pages 255–259, 2004.

[JTH01] W. Jin, A. K. H. Tung, and J. Han. Mining top-n local outliers in large databases. In *7th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 293–298, 2001.

[JTS01] M. F. Jaing, S. S. Tseng, and C. M. Su. Two-phase clustering process for outliers detection. *Pattern Recogn. Lett.*, 22(6-7) :691–700, 2001.

[KH06] Arno J. Knobbe and Eric K. Y. Ho. Maximally informative k-itemsets and their efficient discovery. In *KDD '06 : Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 237–244, New York, NY, USA, 2006. ACM.

[KH07] R. Kwitt and U. Hofmann. Unsupervised anomaly detection in network traffic by means of robust pca. In *International Multi-Conference on Computing in the Global Information Technology*, 2007.

[KN98] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *24rd International Conference on Very Large Data Bases*, pages 392–403, 1998.

[KPWD03] H. Kum, J. Pei, W. Wang, and D. Duncan. ApproxMAP : Approximate mining of consensus sequential patterns. In *Proceedings of SIAM Int. Conf. on Data Mining*, San Francisco, CA, 2003.

[LEK+03] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *3rd SIAM International Conference on Data Mining*, 2003.

[LLC01] Chang-Hung Lee, Cheng-Ru Lin, and Ming-Syan Chen. On mining general temporal association rules in a publication database. pages 337–344, 2001.

[LNWJ03] Yingjiu Li, Peng Ning, X. Sean Wang, and Sushil Jajodia. Discovering calendar-based temporal association rules. *DKE*, 44(2), 2003.

[LS98] W. Lee and S. J. Stolfo. Data mining approaches for intrusion detection. In *7th conference on USENIX Security Symposium*, 1998.

[Luoty] J. Luo. Integrating fuzzy logic with data mining methods for intrusion detection, 1999, Master Thesis, Mississippi State University.

[LX01] W. Lee and D. Xiang. Information-theoretic measures for anomaly detection. In *IEEE Symposium on Security and Privacy*, 2001.

[Mar09] Alice Marascu. *Extraction de motifs séquentiels dans les flux de données*. PhD thesis, Universite de Nice Sophia Antipolis - France, 2009.

[Mas02] Florent Masseglia. *Algorithmes et applications pour l'extraction de motifs sequentiels dans le domaine de la fouille de donnees : de l'incremental au temps reel.* PhD thesis, Universite de versailles St Quentin - France, 2002.

[MCP98a] F. Masseglia, F. Cathala, and P. Poncelet. The PSP Approach for Mining Sequential Patterns. In *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery*, Nantes, France, September 1998.

[MCP98b] F. Masseglia, F. Cathala, and P. Poncelet. The PSP Approach for Mining Sequential Patterns. In *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98)*, pages 176–184, Nantes, France, September 1998.

[MCZH00] S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz. A data mining analysis of rtid alarms. *Computer Networks*, 34, 2000.

[MDLN02] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Discovery and evaluation of aggregate usage profiles for web personalization. *Data Mining and Knowledge Discovery*, 6(1) :61–82, January 2002.

[MM] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th Inter-*

national Conference on Very Large Data Bases, Hong Kong, China, August 2002.

[MM06] Alice Marascu and Florent Masseglia. Extraction de motifs séquentiels dans les flots de données d'usage du Web. In *Actes des 6èmes journées "extraction et gestion des connaissances" (EGC'06)*, Lille, France, 2006.

[MM07] Alice Marascu and Florent Masseglia. Limites d'une approche incrémentale pour la segmentation de séquences dans les flux. In *Atelier Flux de Données à EGC'07*, 2007.

[MPC00] F. Masseglia, P. Poncelet, and R. Cicchetti. An efficient algorithm for web usage mining. *Networking and Information Systems Journal (NIS)*, April 2000.

[MPT99] F. Masseglia, P. Poncelet, and M. Teisseire. Incremental Mining of Sequential Patterns in Large Databases. In *Actes des Journées Bases de Données Avancées (BDA'00)*, Blois, France, Octobre 1999.

[MPT06] Florent Masseglia, Pascal Poncelet, and Maguelonne Teisseire. Peer-to-peer usage analysis : a distributed mining approach. In *AINA (1)*, pages 993–998, 2006.

[MR04] N. Meger and C. Rigotti. Constraint-Based Mining of Episode Rules and Optimal Window Sizes. In *Proc. of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 313–324, Pisa, Italy, September 2004.

[MS03] M. Markou and S. Singh. Novelty detection : a review - part 1 : statistical approaches. *Signal Processing*, 83, 2003.

[MTP01] F. Masseglia, M. Teisseire, and P. Poncelet. Real Time Web Usage Mining : a Heuristic based Distributed Miner (long). In *Web Information Systems Ingineering (WISE'01)*, Kyoto, Japan, December 2001.

[MTT03] F. Masseglia, D. Tanasa, and B. Trousse. Diviser pour découvrir : une méthode d'analyse du comportement de tous les utilisateurs d'un site web. In *Les 19èmes Journées de Bases de Données Avancées*, Lyon, France, 20-24 October 2003.

[MVY05] Shyh-Kwei Chen Michail Vlachos, Kun-Lung Wu and Philip S. Yu. Fast Burst Correlation of Financial Data. In

Knowledge Discovery in Databases : PKDD 2005, pages 422–434, 2005.

[NM03] M. Nakagawa and B. Mobasher. Impact of Site Characteristics on Recommendation Models Based On Association Rules and Sequential Patterns. In *Proceedings of the IJCAI'03 Workshop on Intelligent Techniques for Web Personalization*, Acapulco, Mexico, August 2003.

[ORS98] Banu Ozden, Sridhar Ramaswamy, and Abraham Silberschatz. Cyclic association rules. In *ICDE, Orlando, Florida, USA*, pages 412–421, 1998.

[PES01] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *ACM CSS Workshop on Data Mining Applied to Security*, 2001.

[PKGF03] S. Papadimitriou, H. Kitagawa, P.B. Gibbons, and C. Faloutsos. LOCI : fast outlier detection using the local correlation integral. In *19th International Conference on Data Engineering*, 2003.

[PP07] A. Patcha and J.-M. Park. An overview of anomaly detection techniques : Existing solutions and latest technological trends. *Comput. Networks*, 51, 2007.

[PPK+99] G. Paliouras, C. Papatheodorou, V. Karkaletsis, C.D. Spyropoulos, and P.Tzitziras. From web usage statistics to web usage analysis. In *Proceedings of the IEEE Int. Conf. on Systems Man and Cybernetics*, volume II, pages 159–164, 1999.

[PPK+00] G. Paliouras, C. Papatheodorou, V. Karkaletsis, P.Tzitziras, and C.D. Spyropoulos. Large-scale mining of usage data on web sites. In *AAAI Spring Symposium on Adaptive User Interfaces*, Stanford, California, 2000.

[PSF05] Spiros Papadimitriou, Jimeng Sun, and Christos Faloutsos. Streaming pattern discovery in multiple time-series. In *In VLDB*, pages 697–708, 2005.

[PVKG08] Themis Palpanas, Michail Vlachos, Eamonn J. Keogh, and Dimitrios Gunopulos. Streaming time series summarization using user-defined amnesic functions. *IEEE Trans. Knowl. Data Eng.*, 20(7) :992–1006, 2008.

[Roe98] M. Roesch. SNORT, 1998.

[RPT05] Chedi Raissi, Pascal Poncelet, and Maguelonne Teisseire. Need for SPEED : Mining Sequential Pattens in Data Streams. In *Actes des 21iemes Journees Bases de Donnees Avancees (BDA 2005)*, October 2005.

[RRS00] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. *SIGMOD Records*, 29(2) :427–438, 2000.

[SA96] R. Srikant and R. Agrawal. Mining Sequential Patterns : Generalizations and Performance Improvements. In *Proceedings of the 5th Int. Conf. on Extending Database Technology (EDBT'96)*, pages 3–17, Avignon, France, September 1996.

[Sch94] H Schmidt. Probabilistic part-of-speech tagging using decision trees, revised version, original work. In *the International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK, 1994.

[SDS95] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for computer graphics : A primer, part 1. *IEEE Computer Graphics and Applications*, 15(3) :76–84, 1995.

[SFW99] M. Spiliopoulou, L. C. Faulstich, and K. Winkler. A data miner analyzing the navigational behaviour of web users. In *Proceedings of the Workshop on Machine Learning in User Modelling of the ACAI'99 Int. Conf.*, Creta, Greece, July 1999.

[SLRdATdC09] Alzennyr Da Silva, Yves Lechevallier, Fabrice Rossi, and Francisco de A. T. de Carvalho. Clustering dynamic web usage data. In *Innovative Applications in Data Mining*, pages 71–82. 2009.

[SZ02] Karlton Sequeira and Mohammed Zaki. Admit : anomaly-based data mining for intrusions. In *KDD '02 : Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 386–395, New York, NY, USA, 2002. ACM.

[Tan05] Doru Tanasa. *Web Usage Mining : Contributions to Intersites Logs Preprocessing and Sequential Pattern Extraction with Low Support*. PhD thesis, Universite de Nice Sophia Antipolis - France, 2005.

[TCG] Nele Dexters Toon Calders and Bart Goethals. Mining frequent items in a stream using flexible windows. *Journal of Intelligent Data Analysis*, pages 293–304.

[TCY03] Wei-Guang Teng, Ming-Syan Chen, and Philip S. Yu. A Regression-Based Temporal Pattern Mining Scheme for Data Streams. In *VLDB*, pages 93–104, 2003.

[TT04] D. Tanasa and B. Trousse. Advanced Data Preprocessing for Intersites Web Usage Mining. *IEEE Intelligent Systems*, 19(2) :59–65, April 2004. ISSN 1094-7167.

[VS01] A. Valdes and K. Skinner. Probabilistic alert correlation. In *Recent Advances in Intrusion Detection*, pages 54–68, 2001.

[Web] Webalizer. http ://www.mrunix.net/webalizer/.

[WF06] Raymond Chi-Wing Wong and Ada Wai-Chee Fu. Mining top-k frequent itemsets from data streams. *Data Min. Knowl. Discov.*, 13(2) :193–217, 2006.

[WMG+09] Wei Wang, Florent Masseglia, Thomas Guyet, Rene Quiniou, and Marie-Odile Cordier. A general framework for adaptive and online detection of web attacks. In *WWW*, pages 1141–1142, 2009.

[WZ03] N. Wu and J. Zhang. Factor analysis based anomaly detection. In *IEEE Workshop on Information Assurance*, 2003.

[YBJ04] V. Yegneswaran, P. Barford, and S. Jha. Global intrusion detection in the domino overlay system. In *Network and Distributed Security Symposium*, 2004.

[You95] Randy K. Young. *Wavelet Theory and Its Applications*. Kluwer Academic Publishers Group, 1995.

[YZS05] Jin Soung Yoo, Pusheng Zhang, and Shashi Shekhar. Mining time-profiled associations : An extended abstract. In *PAKDD, Hanoi, Vietnam, May 18-20*, pages 136–142, 2005.

[ZHH02] J. Zhu, J. Hong, and J. G. Hughes. Using Markov Chains for Link Prediction in Adaptive Web Sites. In *Proceedings of Soft-Ware 2002 : First Int. Conf. on Computing in an Imperfect World*, pages 60–73, Belfast, UK, April 2002.

[ZKS07] S. Zhong, T. M. Khoshgoftaar, and N. Seliya. Clustering-based network intrusion detection. *International Journal of Reliability, Quality and Safety Engineering*, 14, 2007.

[ZRL96] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch : An efficient data clustering method for very large databases. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD Int. Conf. on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 103–114. ACM Press, 1996.

[ZS02] Yunyue Zhu and Dennis Shasha. Statstream : statistical monitoring of thousands of data streams in real time. In *VLDB '02 : Proceedings of the 28th international conference on Very Large Data Bases*, pages 358–369, Hong Kong, China, 2002. VLDB Endowment.

[ZS03] Yunyue Zhu and Dennis Shasha. Efficient elastic burst detection in data streams. In *KDD '03 : Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 336–345, 2003.

[ZXH98] Osmar R. Zaiane, Man Xin, and Jiawei Han. Discovering web access patterns and trends by applying OLAP and data mining technology on web logs. In *Advances in Digital Libraries*, pages 19–29, 1998.