Résumé de flux de données ditribués

Raja Chiky

TELECOM ParisTech

École Doctorale d'Informatique, Télécommunications et Électronique de Paris

Thèse

Résumé de flux de données distribués

présentée et soutenue publiquement le 23 janvier 2009

pour l'obtention du

Doctorat de l'École Nationale Supérieure des Télécommunications

spécialité : Informatique et Réseaux

par

Raja CHIKY

Composition du jury

Président :	Geneviève Jomier	Université Paris Dauphine
Rapporteurs :	Hervé Cardot	Université de Bourgogne
	Bruno Defude	Télécom & Management SudParis
Examinateurs :	Laurent Decreusefond	Télécom ParisTech
	Alain Dessertaine	EDF R&D
Directeur de thèse :	Georges Hébrail	Télécom ParisTech

A mes parents,
A tous ceux qui comptent dans ma vie,

Remerciements

Ce travail n'aurait pu être sans l'accord, le soutien et l'aide de plusieurs personnes. Je tiens à exprimer ma profonde gratitude et mes remerciements à :

Mon directeur de thèse Georges Hebrail, professeur à Télécom ParisTech, pour sa collaboration, sa disponibilité, ses informations fructueuses, son soutien tout au long de la réalisation de ce travail et surtout pour sa sympathie et sa gentillesse. Bien que débordé par le travail, il a toujours trouvé le temps pour m'encadrer, et me guider durant la thèse.

Hervé Cardot, professeur à l'université de Bourgogne et Bruno Defude, professeur à Télécom et Management SudParis qui m'ont fait l'honneur de prendre connaissance de ce travail et d'en être rapporteurs. Je les remercie pour leurs conseils et leurs suggestions qui ont permis l'amélioration de ce manuscrit.

Geneviève Jomier, professeur à l'université de Dauphine pour l'honneur qu'elle m'a fait d'avoir accepté de présider le jury de ma thèse.

Laurent Decreusefond, professeur à Télécom ParisTech, pour son aide précieuse en mathématiques ainsi que pour sa participation en tant que membre du jury.

Alain Dessertaine pour nos fréquentes discussions sur de nombreux sujets, pour son implication dans mes travaux et le temps consacré à la relecture du manuscrit.

Marie-Luce Picard pour avoir assuré le suivi de la thèse à EDF. Ainsi que Françoise Guisnel et Jérôme Cubillé pour leurs participations aux discussions autour de ma problématique de thèse. Je remercie aussi tous les membres du groupe SOAD pour leur accueil et tous les moments partagés.

Les membres (passé et présent) du département Informatique et Réseaux, dirigé par Michel Riguidel. Tout particulièrement, les non permanents : Irfan 'le truand', Bechir 'le bon', Olivier 'la brute', Billel 'le bosseur', Nesrine 'la stressée', Nora 'la coquette', Hoa 'wikipedia', Gilles 'l'indispensable', Khalil 'le virtuel', Khaled 'le souriant'. Les permanents : Céline Bizart, Hayette Soussou, Jean-Louis Dessalles, Olivier Hudry, Serge Gadret, Hassane Aissaoui, Fabrice Rossi, Annie danzart, Christine Potier et tous ceux qui ont permis à notre département de devenir un lieu sympathique et convivial.

Ma famille pour leur confiance inconditionnelle, leur affection et leurs encouragements.

Enfin, merci à toutes les personnes qui m'ont aidée de près ou de loin à la réalisation de ce projet.

J'espère que ce travail réalisé grâce à vous sera à la hauteur de la confiance et de l'apport que vous m'avez donné à travers nos différents contacts.

Résumé

Ces dernières années, sont apparues de nombreuses applications, utilisant des données en nombre potentiellement illimité, provenant de façon continue de capteurs distribués afin d'alimenter un serveur central. Les données sont utilisées à des fins de surveillance, de supervision, de déclenchement d'alarmes en temps réel, ou plus généralement à la production de synthèses d'aide à la décision à partir de plusieurs flux. Le volume des données collectées est généralement trop grand pour être entièrement stocké. Les systèmes de gestion de flux de données (SGFD) permettent de gérer facilement, et de façon générique les flux de données : les données sont traitées au fil de l'eau sans les archiver. Toutefois, dans certaines applications, on ne souhaite pas perdre complètement l'ensemble des flux de données afin de pouvoir analyser les données du passé et du présent. Il faut donc prévoir un stockage de l'historique du flux.

Nous considérons dans cette thèse, un environnement distribué décrit par une collection de plusieurs capteurs distants qui envoient des flux de données numériques et unidimensionnelles à un serveur central unique. Ce dernier a un espace de stockage limité mais doit calculer des agrégats, comme des sommes ou des moyennes, à partir des données de tout sous-ensemble de capteurs et sur un large horizon temporel. Deux approches sont étudiées pour construire l'historique des flux de données :(1) Echantillonnage spatial en ne considérant qu'un échantillon aléatoire des sources qu'on observe dans le temps ; (2) Echantillonnage temporel en considérant toutes les sources mais en échantillonnant les instants observés de chaque capteur. Nous proposons une méthode générique et optimisée de construction de résumés à partir de flux de données distribués : A partir des flux de données observés à une période de temps $t-1$, nous déterminons un modèle de collecte de données à appliquer aux capteurs de la période t.

Le calcul des agrégats se base sur l'inférence statistique dans le cas de l'échantillonnage spatial et sur l'interpolation dans le cas de l'échantillonnage temporel. A notre connaissance, il n'existe pas de méthodes d'interpolation qui estiment les erreurs à tout instant et qui prennent en compte le flux de données ou courbe à interpoler et son intégrale. Nous proposons donc deux approches : la première se base sur le passé des courbes pour l'interpolation (approche naive) ; et la seconde utilise à un processus stochastique pour modéliser l'évolution des courbes (approche stochastique).

Mots-clés: Flux de données distribués, Echantillonnage adaptatif, capteurs, interpolation, courbes de charge

Abstract

Remote sensing enables data collection from multiple sources in a unique central server for a large wide of applications which are typically monitoring and system supervision, transmitting alarms in real time and more generally producing synthesis to help in business decision. The volume of data collected is generally too big to be stored entirely. Data Stream Management Systems are generic tools for processing data streams : data elements arrive on-line and stay only for a limited time period in memory. For many stream processing applications, one may not wish to completely lose the entire stream data for analyzing 'past' and 'present'data. Thus a stream processing system must also provide a stored historical data.

In this thesis, we consider a distributed computing environment, describing a collection of multiple remote *sensors* that feed a unique *central server* with numeric and uni-dimensional data streams (also called curves). The central server has a limited memory but should be able to compute aggregated value of any subset of the stream sources from a large time horizon including old and new data streams. Two approaches are studied to reduce the size of data : (1) spatial sampling only consider a random sample of the sources observed at every instant ; (2) temporal sampling consider all sources but samples the instants to be stored. In this thesis, we propose a new approach for summarizing temporally a set of distributed data streams : From the observation of what is happening during a period $t-1$, we determine a data collection model to apply to the sensors for period t.

The computation of aggregates involves statistical inference in the case of spatial sampling and interpolation in the case of temporal sampling. To the best of our knowledge, there is no method for estimating interpolation errors at each timestamp that would take into account some curve features such as the knowledge of the integral of the curve during the period. We propose two approaches : one uses the past of the data curve (naive approach) and the other uses a stochastic process for interpolation (stochastic approach).

Keywords: Distributed Data Streams, adaptative sampling, sensors, interpolation, load curves

Table des matières

Liste des illustrations

Liste des tableaux

Chapitre 1

Introduction et Motivation

Présentation de la problématique

Ces dernières années ont vu émerger un grand nombre d'applications utilisant des données en nombre potentiellement infini, produites de façon continue par de multiples capteurs. Parmi ces applications, citons par exemple les réseaux informatiques, les systèmes de transactions bancaires et financières, les compteurs relevant la consommation électrique ou les réseaux de capteurs. Ces systèmes utilisent généralement un ensemble de capteurs, organisés ou non en réseau, qui constituent le système d'acquisition. Ces capteurs envoient des mesures en temps réel et à fréquence variable, sous forme de flux de données, vers un système centralisé. Golab [69] définit un flux de données comme une séquence d'enregistrements continue, ordonnée (implicitement par temps d'arrivée ou explicitement par une estampille temporelle de production à la source), arrivant en temps réel. Les systèmes centralisés ne peuvent pas faire face à la volumétrie et au taux d'arrivée des flux de données car il n'est possible ni de contrôler l'ordre d'arrivée des enregistrements, ni de stocker localement un flux dans sa globalité. De nouveaux outils appelés Systèmes de Gestion de Flux de Données (SGFD) ont fait leur apparition sur le marché qui permettent de traiter à la volée (en ligne) les flux de données provenant des capteurs, mais à des seules fins de supervision. Or, dans plusieurs applications, il est nécessaire de garder une trace des flux de données afin d'effectuer des traitements hors-ligne. Comme il est inconcevable de stocker la totalité des flux de données infinis, la solution consiste à utiliser des techniques d'approximation dont la qualité du résultat sera ajustée en fonction des ressources disponibles.

Le travail mené dans le cadre de cette thèse consiste à étudier une solution à la problématique de volumétrie des données provenant de multiples sources en proposant des techniques de résumé adaptées aux cas des flux de données arrivant en temps réel. Nous utilisons aussi dans la suite de ce manuscrit le terme courbes pour qualifier les flux de données.

Résumé de flux de données distribués

D'un point de vue général, Csernel [47] définit un résumé comme une acquisition d'information compacte à partir d'un flux de données dans son intégralité temporelle et sans envisager de traitement sur le résumé a priori. Plusieurs techniques visant à résumer les données d'un flux individuel ont vu le jour ces dernières années. Dans le cadre de cette thèse, nous travaillons avec plusieurs sources de flux de données i.e. dans environnement distribué. Notre objectif est de constituer un résumé généraliste tel que défini par [47] mais en prenant en compte la totalité des flux de données. Ce résumé sera collecté par un serveur central à partir des capteurs de données.

Une approche pouvant naturellement être utilisée pour réduire la volumétrie des données à partir de capteurs distribués est de construire un panel. Un panel est constitué à partir d'une population finie (l'ensemble des capteurs) échantillonnée selon une stratégie qui permet d'exploiter au mieux les informations disponibles sur les différents capteurs. Les panels sont utiles pour estimer un paramètre qui fait explicitement intervenir les évolutions individuelles d'une donnée à mesurer. L'intérêt des panels apparaît naturellement dans le traitement des flux de données, puisque ces derniers sont constitués d'informations dans le temps. Nous appelons cette approche *échantillonnage spatial*.

Une seconde approche visant à réduire la volumétrie des données est de limiter le volume de données envoyé par chaque capteur au système central. Ainsi, les capteurs construisent dans le temps un résumé des données qu'ils envoient au système central qui a la charge de les fusionner. Nous qualifions ce type de construction de résumé d'*échantillonnage temporel*. Dans cette approche, la construction du résumé doit respecter un certain nombre de contraintes liées à la nature des flux de données. Premièrement, le résumé doit être constitué au fil de l'eau, avec un temps de traitement par élément faible, pour pouvoir faire face au taux d'arrivée des éléments. Deuxièmement, les résumés doivent être de taille limitée pour permettre leur stockage au niveau du système central. Troisièmement les résumés doivent être continuellement mis à jour afin de gérer le caractère continu du flux de données. Pour faire face à cet ensemble de contraintes, nous proposons un outil générique de résumé de flux de données distribués, qui utilise des techniques de résumés sur flux de données telles que l'échantillonnage régulier (une fréquence d'échantillonnage est définie pour chaque capteur), la segmentation des courbes par programmation dynamique et la compression des courbes par ondelettes. Cet outil permet de distribuer de façon optimisée les niveaux de résumés à appliquer à chaque capteur. Nous avons démontré l'efficacité de nos algorithmes en les appliquant à des jeux de données réels de consommation électrique.

Nous proposons aussi une stratégie dite *échantillonnage spatio-temporelle* qui consiste à combiner l'échantillonnage temporel et l'échantillonnage spatial, dans l'objectif d'obtenir des estimateurs des flux de données agrégés de qualité acceptable (avec des erreurs d'interpolation et/ou d'échantillonnage faibles).

Estimation des flux de données résumés

Si, dans le cas de l'échantillonnage spatial, il existe des estimateurs dans la théorie des sondages qui permettent d'effectuer des inférences statistiques afin d'estimer l'agrégation des courbes (la courbe globale par exemple) ainsi que les erreurs d'estimation, dans le cas de l'échantillonnage temporel, les courbes récupérées par le serveur central sont résumées à des instants différents. Il est donc nécessaire de reconstruire les courbes afin de les agréger. Pour ce faire, nous proposons deux méthodes d'interpolation des flux de données résumés par l'approche temporelle en prenant en compte l'intégrale connue des flux. Une des méthodes proposées se base sur le passé d'un flux de données afin de trouver une bonne interpolation de la courbe résumée, nous l'appelons *approche naive*. La seconde méthode se base sur une modélisation du comportement du flux de données afin de déterminer une interpolation qui respecte la contrainte de l'intégrale, nous l'appelons *approche stochastique*. Ce nom est du au processus stochastique utilisé dans la modélisation : le mouvement brownien géométrique comme modèle de l'évolution du flux de données. A notre connaissance, aucune des méthodes d'interpolation ou de régression existantes dans la littérature ne permet de prendre en compte la courbe et son intégrale. De plus, il est de coutume quand on applique une méthode d'interpolation (ou de régression) de prendre comme hypothèse que l'erreur commise est équirépartie sur toute la période sur laquelle on interpole, c'est ce qu'on appelle l'homoscédasticité. Or, avec l'échantillonnage temporel, nous récupérons des éléments des courbes à différents instants pour lesquels nous connaissons leurs valeurs, et par conséquent l'erreur d'interpolation est nulle. Par souci de réduire les erreurs d'interpolation quand on agrège les courbes, nous levons l'hypothèse de l'homoscédasticité. Ainsi, les méthodes que nous proposons fournissent une interpolation de la courbe ainsi qu'une estimation de l'erreur à chaque instant où une interpolation est nécessaire. L'erreur est donc nulle aux instants sélectionnés par l'échantillonnage temporel.

Application

Après avoir défini et développé les deux approches d'échantillonnage (temporel et spatial), nous proposons de les comparer au travers d'un cas réel : la mesure des consommations électriques des abonnés au réseau de distribution d'électricité d'EDF. Aujourd'hui, EDF maintient un panel « physique » de clients décrits par leurs courbes de consommation électrique et d'autres informations telles que des réponses à des enquêtes, les équipements électriques utilisés, le type de tarification, etc. Dans quelques années, EDF envisage de déployer des compteurs électroniques qui permettront de relever à distance des données de consommation électrique à des intervalles de temps très fin (à partir de la seconde) chez l'ensemble des abonnés. Ce nouveau système générera des flux de données volumineuses qu'il n'est pas concevable de stocker en totalité. Une approche de type résumé serait donc adaptéee pour traiter ces flux de données. Nous avons étudié notre approche d'échantillonnage temporel sur un certain nombre de compteurs électriques et nous l'avons comparée

à un échantillonnage spatial. La comparaison s'est faite sur la base de réponse aux requêtes d'agrégation, et en particulier afin d'estimer la courbe de consommation globale de l'ensemble ou d'un sous-ensemble de capteurs.

Plan de la thèse

Ce document est composé de trois parties principales :

1. Etat de l'art du domaine de gestion de flux de données. Le chapitre 2 est un chapitre préliminaire où nous introduisons la notion de flux de données. Ce domaine a récemment fait l'objet de nombreuses études, en raison d'une part du nombre important d'applications émergentes qui manipulent des flux et d'autre part, de l'impossibilité d'utiliser directement les systèmes de gestion de bases de données pour traiter ce type d'applications. Nous définissons les notions liées aux flux de données, citons des exemples d'applications faisant intervenir les flux de données, et nous énumérons les enjeux qu'un système doit relever afin de traiter ce type de données. Le chapitre 3 dresse un panorama des méthodes d'approximation d'un flux de données individuel. De nombreuses structures de résumés ont été développées ces dernières années comme les sketches, l'échantillonnage, les ondelettes et les histogrammes répondant à plusieurs types de traitements, capables de résumer de manière significative de grandes quantités de données. De plus, ces méthodes ne nécessitent qu'une passe sur les éléments ce qui les rend applicables dans le cas des flux de données. Plus particulièrement, elles peuvent s'insérer dans notre outil générique développé afin de résumer les flux de données distribués. Au chapitre 4, nous citons les travaux de recherche menés pour développer des algorithmes de résumés des données au fil de l'eau appliqués à des flux de données distribués et que nous avons divisé en deux grandes catégories. Ceux qui échantillonnent les sources de flux de données (*échantillonnage spatial*), et ceux qui résument les données de chaque capteur (*échantillonnage temporel*). Nous avons présenté une application particulière à la problématique de la thèse au chapitre 5 ainsi que les jeux de données utilisés pour évaluer nos travaux.
2. Proposition d'une nouvelle approche d'échantillonnage temporel. Après avoir défini les différentes méthodes existantes de résumé de flux de données, nous présentons une contribution dans ce sens dans le chapitre 6 qui permet de rassembler ces méthodes en un outil générique qui permet d'affecter des niveaux de résumé par optimisation aux capteurs afin de réduire le plus possible les erreurs commises ainsi que la surcharge du serveur central. Cet outil s'adapte continuellement au contenu des flux de données. Ainsi, la technique de résumé adoptée par chaque capteur dépend des variations des données dans le flux et est choisie parmi plusieurs approches. Nous comparons cette méthode d'échantillonnage temporel avec l'échantillonnage spatial au chapitre 7 afin de répondre à des requêtes d'agrégation des données provenant de la totalité des capteurs. Nous proposons aussi une approche d'échantillonnage spatial de

capteurs distribués couplée à un échantillonnage temporel que nous appelons *échantillonnage spatio-temporel* et que nous comparons aux deux approches précédentes (purement temporel et purement spatial).

3. Reconstruction des flux de données distribués et estimation des erreurs. Nous proposons dans le chapitre 8, de comparer les deux approches spatiale et temporelle dans un contexte de petit domaine. En effet, il arrive souvent que l'on s'intéresse à des totaux ou des moyennes ne concernant pas l'ensemble des capteurs, mais seulement un sous ensemble de capteurs appelé domaine. Par exemple, on peut chercher à calculer la courbe de consommation globale en électricité des habitants de Paris jour par jour. Si on connait a priori les requêtes pouvant être posées sur les domaines, on choisit évidemment de ne s'intéresser qu'aux capteurs pouvant répondre à ces requêtes. On peut appliquer par exemple un échantillonnage stratifié dans le cas du résumé spatial ou affecter des niveaux de résumé conditionnés à l'intérêt des capteurs vis à vis des requêtes dans le cas du résumé temporel. Malheureusement, ce n'est pas toujours le cas, et il faut donc d'abord effectuer un résumé et ensuite s'intéresser aux différents domaines. Afin de comparer les deux approches dans ce contexte, nous proposons deux méthodes d'interpolation des flux de données résumés par l'approche temporelle en prenant en compte l'intégrale connue des flux.

Finalement, le chapitre 9 conclut ce travail et donne quelques perspectives de recherche.

Chapitre 2

Systèmes de gestion de flux de données

Sommaire

2.1 Introduction

De nos jours, l'avancée permanente des nouvelles technologies permet de collecter continuellement de plus en plus d'informations à partir d'une seule ou de multiples sources distantes. Les applications actuelles des approches de traitement de ces informations portent surtout sur la supervision de systèmes, sur le déclenchement d'alarmes en temps réel, ou plus généralement sur la production de synthèses d'aide à la décision à partir de plusieurs sources de données. L'approche la plus immédiate pour le traitement des données est d'utiliser un SGBD (Système de Gestion de Bases de Données) traditionnel qui assure la consultation de la globalité des données stockées par le traitement et l'évaluation des requêtes. Beaucoup de solutions industrielles ont choisi cette approche. Cependant, les méthodes utilisées actuellement dans les SGBD traditionnels ne sont pas ou peu adaptées à de nombreux cas pour

gérer les données provenant de sources de flux de données (c'est-à-dire des données en circulation avec des débits en général importants tels que les consommations électriques). Premièrement, ces sources peuvent générer un grand volume de données qui provoquent aussi un grand nombre de mises à jour de la base. Deuxièmement, dans les SGBD actuels les requêtes sont évaluées une fois quand elles sont soumises. Cette approche n'est pas appropriée pour les flux de données puisque celles-ci changent en temps réel et une évaluation de manière continue des requêtes serait plus adaptée. Troisièmement, le caractère "push" de la génération des données par les sources de flux n'est pas compatible avec les SGBD où l'entrée des données est de type "pull" (réalisée par des commandes liées au SGBD). Dernièrement, une construction dynamique du plan d'exécution est essentielle. Elle doit être adaptable et tolérante aux pannes puisque les sources des flux et leurs données sont très vulnérables aux pannes, aux pertes d'informations et aux imprécisions.

Pour faire face à ces spécifités, la notion de Système de Gestion de Flux de Données (SGFD) ou DSMS (en anglais pour Data Stream Management System) a été proposée. Les SGFD définissent un flux comme une séquence infinie de données structurées sous forme de n-uplets (enregistrements) arrivant sous forme continue et ordonnée. L'ordre est généralement défini par rapport au temps soit implicitement par temps d'arrivée, soit explicitement grâce à un attribut correspondant au timestamp. Les données ne sont pas disponibles par accès aléatoire sur disque. Les données traitées par les SGFD sont des flux transitoires et temporairement stockées dans la mémoire centrale, en général dans une structure de file d'attente. Donc, la taille des données qui peuvent être stockées étant très limitée, les données sont simplement éliminées après leur traitement ou archivées dans des structures de résumés. Les SGFD proposent des mécanismes pour gérer en temps réel des flux infinis. Ils permettent de formuler des requêtes continues qui s'évaluent au fur et à mesure sur un flux ou sur des fenêtres qui sont des sous ensembles finis du flux ([17], [63], [69], [93]).

Le domaine des flux de données a donc été le sujet d'un intérêt grandissant de la part de différentes communautés (base de données, analyse de données, réseaux) aussi bien du côté industriel qu'universitaire. Un intérêt qui reflète le nombre croissant d'applications et de systèmes industriels confrontés à des flux de données que l'on souhaite mesurer ou contrôler d'une façon ou d'une autre. C'est donc un nouveau mode de traitement des données qui émerge, avec toutes les questions et les particularités qu'il peut apporter. La nature imprévisible du taux d'arrivée des n-uplets implique que le système doit être capable de s'adapter dynamiquement à des cas de surcharge. En particulier, si le système n'est pas assez rapide pour traiter tous les enregistrements d'un flux, plusieurs solutions peuvent être envisagées. Il est possible d'utiliser des buffers, et en cas de débordement de ces buffers, il faut ignorer certains n-uplets, en procédant par exemple par échantillonnage ou par Load Shedding ([19], [106]). Quelques systèmes utilisent une seconde approche correspondant à des résumés de l'information (sketches ou synopsis [93])). Toutes ces nouvelles notions seront traitées dans les prochaines sections de ce mémoire.

2.2 Domaines d'application

Les domaines d'applications que nous avons prioritairement identifiés offrent un panorama des enjeux de la gestion des flux de données au fil de l'eau : la surveillance de données de capteurs, l'analyse du trafic dans un réseau informatique et le traitement des données financières.

Les données de capteurs peuvent être modélisées par des flux de données. Les capteurs sont utilisés en général à des fins de contrôle (monitoring) dans des domaines comme la consommation électrique, le trafic routier, la météorologie ou encore la surveillance de procédés de fabrication. Bien que les requêtes dépendent essentiellement du genre d'application, on retrouve un certain nombre de requêtes typiques, comme par exemple : Activer une alerte si des capteurs d'une même zone dépassent une valeur donnée. Pour un suivi de consommation électrique, une alerte peut être envoyée à un utilisateur en cas de dépassement d'un seuil. On peut aussi analyser le flux produit par des capteurs de trafic routier pour détecter des situations pouvant mener à des embouteillages, de façon à informer les usagers ou à mettre en place des redirections.

On retrouve des besoins similaires dans l'analyse du trafic dans les réseaux informatiques. On peut vouloir par exemple détecter les situations critiques (la congestion et le refus de service) ou encore analyser l'utilisation de la bande passante ou les adresses IP les plus actives. Un exemple de requête typique du domaine peut concerner le total de la bande-passante utilisée par chaque paire source-destination groupé par le type de protocole ou le masque de sous-réseau. Toujours dans le domaine des télécommunications, un autre exemple est le traitement des tickets d'appels téléphoniques afin d'analyser les habitudes des utilisateurs ou d'appliquer des facturations spéciales à un certain nombre de clients. On peut citer comme exemple de requêtes : déterminer pour chaque client ses trois appels les plus longs dans le dernier mois.

Enfin, une grande famille d'applications est celle de l'analyse financière. En effet, il est assez naturel de modéliser l'ensemble des cours boursiers comme un flux de données arrivant à l'application. L'analyse en ligne des cours des actions implique la découverte de corrélations et de variations des prix, l'identification des tendances et des opportunités d'arbitrage, la détection des fraudes, etc. Un exemple de requête peut concerner les actions à haute volatilité et gros volumes. Par exemple, les actions dont le volume d'échanges a augmenté de plus de 300% dans les 5 dernières minutes et dont le prix de vente a varié de plus de 3%.

2.3 Définitions des concepts de base des flux de données

Une définition d' un flux de données a été donnée par Golab et Özsu dans leur article [69] : *A data stream is a real-time, continuous, ordered (implicitly by arrival time or explicitly by timestamp) sequence of items. It is impossible to control the order in which items arrive, nor is it feasible to locally store a stream in its entirety. Likewise, queries over streams run continuously over a period of time and incre-*

mentally return new results as new data arrive. These are known as long-running, continuous, standing, and persistent queries. Ce qui se traduit par : « Un flux de donnée est une séquence d'items continue, ordonnée (implicitement par temps d'arrivée dans le Système de Gestion de Flux de Données, ou explicitement par timestamp de production à la source), arrivant en temps réel. Il est impossible de contrôler l'ordre dans lequel arrivent les items, et il est impossible de stocker localement un flux dans sa globalité. Les requêtes sur un flux de données sont actives continuellement sur une période de temps et retournent incrémentalement de nouveaux résultats lorsque des nouvelles données arrivent. Ces requêtes sont dites persistantes, continues et de longue durée d'exécution ».

La structure d'un flux peut être assimilée à celle d'une table relationnelle comme dans les SGBD traditionnels et les items à des n-uplets arrivant en ligne. La structure du flux est représentée par un schéma comprenant le nom des champs du n-uplet et leur type. Un exemple de flux de données généré par un compteur électrique est illustré dans la figure 2.1, le compteur produit chaque minute des informations concernant la consommation électrique.

Timestamp	Puis. A (kW)	Puis. R(kVar)	U1(V)	I1(A)
...	...	...	...	...
16/12/2006-17 :26	5,374	0,498	233,29	23
16/12/2006-17 :27	5,388	0,502	233,74	23
16/12/2006-17 :28	3,666	0,528	235,68	15,8
16/12/2006-17 :29	3,52	0,522	235,02	15
...	...	...	...	...

TABLE 2.1 – Exemple de flux de données

De la même manière que dans les SGBD traditionnels, les opérateurs effectuent des traitements à partir des données et produisent des résultats en sortie, qui sont transmis à d'autres opérateurs ou directement envoyés à l'utilisateur. Une requête portant sur le filtrage de données d'un flux potentiellement infini ne pose pas réellement de problème. Par contre si une requête contient des opérateurs dits « bloquants », il faut réduire le flux infini à une séquence finie. C'est le cas pour un opérateur de jointure : le calcul d'une jointure sur des éléments défilants qui disparaissent au fur et à mesure rend les choses difficiles. Pour effectuer de telles opérations, les SGFD utilisent une portion du flux sur lequel portera un traitement et qui se nomme une *fenêtre*. Cette dernière peut être classée dans une des catégories suivantes :

– Physique vs. logique : les fenêtres physiques (appelées aussi temporelles) sont définies au moyen d'un intervalle de temps comme par exemple une fenêtre à partir du 01/01/2008 au 31/05/2008, alors que les fenêtres logiques (appelées aussi séquentielles) sont définies en terme de nombre d 'items, comme par exemple une fenêtre du 10^{eme} élément au 100^{eme}. Le fait de distinguer ces deux types de fenêtres a une grande importance dans la conception des algorithmes. En effet, la taille de la fenêtre logique est connue a priori, alors que celle de

la fenêtre physique est connue a posteriori si le débit du flux de données est variable.

– Fixe vs. glissante : Une fenêtre fixe est une portion du flux dont les bornes sont fixées comme par exemple une fenêtre entre le 1^{er} mars et le 1^{er} avril 2008. Paradoxalement, les bornes d'une fenêtre glissante évoluent avec le temps. Par exemple : les 10 derniers éléments du flux ou les éléments arrivés il y a moins de 15 secondes. Nous spécifions sa taille ainsi que son pas de décalage dans le temps (ou en nombre d'éléments). Si le pas de décalage est inférieur à la taille de la fenêtre, on parle de fenêtre purement *glissante*. Si le pas de décalage est égal à la taille de la fenêtre, on parle de fenêtre *sautante*. Enfin, si le pas de décalage est supérieur à la taille de la fenêtre, on parle de fenêtre *bondissante*. La figure 2.1 (extraite de [61]) montre la différence entre ces trois types de fenêtres. Dans quelques références, on parle d'une troisième catégorie : fenêtre *point de repère* (en anglais Landmark), quand une seule borne de la fenêtre se déplace dans le flux et l'autre reste fixe, une fenêtre entre le 1^{er} mars et le temps courant en est un exemple.

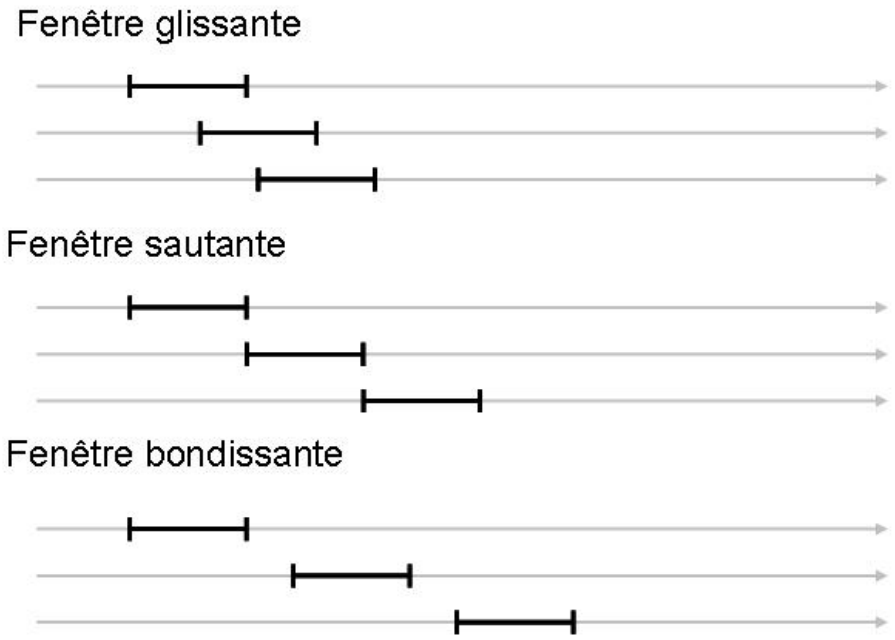

FIGURE 2.1 – Différents types de fenêtres gissantes

2.4 Traitement de flux de données : Enjeux

En présence de flux, on peut choisir d'utiliser un Système de Gestion de Bases de Données pour alimenter un entrepôt de données. Celui-ci offre la possibilité de regrouper des données et de les exploiter [80]. Pour alimenter l'entrepôt de données, on utilise généralement un ETL (Extract, Transform and Load), outil décrivant les données, leur provenance et les transformations effectuées. Il permet d'agréger, de classifier,

de normaliser, de nettoyer les données extraites et de les charger dans un entrepôt de données. L'alimentation d'un entrepôt de données se fait de façon périodique suivant une périodicité définie par l'administrateur. Des décisions d'agrégation ou d'échantillonnage des données sont faites à l'avance. L'analyse et le traitement des données se font traditionnellement dans les entrepôts de données où elles sont agrégées. L'apparition des flux de données s'accompagne de quelques difficultés liées essentiellement à la taille, à la rapidité d'arrivée et à l'éventuelle distribution des données et des structures manipulées. Ceci pousse à revisiter les technologies d'entrepôts de données et d'analyse en ligne. En effet, l'architecture doit s'adapter à la distribution des données, à leur volumétrie croissante et à leur taux d'arrivée. L'acquisition, la modélisation, le nettoyage, l'interrogation et l'analyse doivent être étendus aux flux de données. En lieu et place d'un SGBD, un Système de Gestion des Flux de Données (SGFD) doit être considéré. En plus des fonctionnalités classiques d'un SGBD, celui-ci autorise une gestion et une analyse à la volée et en temps réel de données se présentant continuellement. Plusieurs systèmes prototypes ont été développés, certains avec une vocation généraliste, alors que d'autres sont destinés à un type particulier d'application. De plus, les offres commerciales issues de ces recherches commencent à émerger. Nous présenterons quelques uns de ces SGFD dans la section 2.7. Un système gérant un flux de données doit disposer d'un certain nombre de caractéristiques [104; 69] lui permettant de répondre aux besoins des applications citées précédemment :

Résistance au flux En raison des contraintes de stockage et de performance, les données doivent être traitées à la volée. Les algorithmes de flux en ligne sont restreints à ne faire qu'un seul passage sur les données et tout traitement doit pouvoir se passer d'un stockage impératif des données.

Intégrité des données Le système de gestion de flux de données doit s'assurer de l'intégrité et la disponibilité des données à tout instant en dépit des éventuelles pannes. Le système doit aussi gérer les "imperfections" des données qui arrivent en ligne. Il doit prendre en compte d'éventuels retards, données manquantes ou données inhabituelles. Dans un système de gestion de flux, il est par exemple impossible d'attendre indéfiniment l'arrivée d'un n-uplet. Un délai maximal d'exécution doit être adjoint à toute opération potentiellement bloquante.

Traitement des requêtes continues Les requêtes traitant les données doivent s'évaluer continuellement afin de prendre en compte les nouvelles valeurs s'ajoutant au flux, et les réponses sont créées aussi en flux. Pour la formulation des requêtes, l'approche la plus adoptée est d'utiliser le modèle relationnel et de construire des requêtes avec un langage similaire à SQL. Le modèle de données et la sémantique des requêtes doivent permettre les traitements basés sur le temps ou l'ordre des données. La sémantique du SQL standard est étendue en introduisant la notion de fenêtre (portion finie d'un flux) et des opérateurs spécifiques au traitement de flux. De plus, on peut attendre d'un système qu'il permette à l'utilisateur de définir ses propres opérateurs.

Traitement des données statiques et continues Dans de nombreuses applications, il peut être intéressant de comparer des données récentes avec des données historiques. Par exemple, dans les applications de détection de fraude, identifier un comportement « inhabituel » nécessite par définition d'avoir une base de données statique correspondant aux comportements « habituels ». Ainsi les données récentes et les données historiques doivent être traitées ensemble dans la même application. Pour ce faire, le système de gestion de flux de données doit pouvoir mêler les données statiques et les données dynamiques. Il doit autoriser le stockage, l'accès et la modification de données statiques et permettre leur combinaison avec des données dynamiques.

Passage à l'échelle Il est recommandé que le système fonctionne de manière distribuée. Les applications sont alors réparties sur plusieurs machines afin de passer à l'échelle. Le système doit également être capable de répartir le traitement sur plusieurs processeurs et de supporter le multithreading, tout cela de manière automatique, transparente et efficace.

Traitement de la volumétrie Le système répond instantanément. Il fournit ainsi des réponses en temps réel en faisant face à un gros volume de données, ce qui nécessite un moteur hautement optimisé. L'exécution partagée de nombreuses requêtes continues est requise pour soulager la charge du système afin d'assurer ses fonctions même en cas de grosse volumétrie. De plus, l'incapacité de stocker un flux entier suggère l'utilisation de structures de résumés (approximation). Par conséquent, des requêtes sur des résumés ne retournent pas de réponses exactes.

2.5 Types de requêtes sur les flux

Comme pour les SGBD, un SGFD doit être capable d'interroger et analyser les données à travers des requêtes posées. En plus des requêtes *ponctuelles* évaluées une seule fois, les SGFD traitent des requêtes dites *continues*, qui sont actives continuellement sur une période de temps. Le résultat des requêtes continues est soit stocké avec des mises-à-jour à l'arrivée de nouveaux éléments, soit restitué à l'utilisateur sous forme de flux.

On peut aussi distinguer dans les SGFD deux types de requêtes : *prédéfinies* et *ad-hoc*. Les requêtes prédéfinies sont fournies au SGFD avant même l'arrivée des données, elles sont généralement continues. Une requête ponctuelle peut elle aussi être prédéfinie. Quant aux requêtes ad-hoc, elles sont posées en ligne après l'arrivée des éléments. Elles peuvent être continues ou ponctuelles. Ce type de requêtes complique la conception des SGFD pour deux raisons. D'une part, comme elles ne sont pas connues à l'avance, elles ne sont pas introduites dans un processus d'optimisation et d'identification de sous-expressions communes entre les requêtes. D'autre part, la réponse à une requête ad-hoc peut requérir l'archivage d'anciens éléments.

On trouve dans [69] une liste de requêtes continues sur les flux de données communes à de nombreux besoins d'analyse de flux. Il est nécessaire de garder à l'esprit que des applications spécifiques pourraient avoir des besoins supplémentaires :

- Sélection : appliquer des filtres complexes à des éléments du flux.
- Agrégation [56; 65] : effectuer des agrégats pour capturer les tendances (comparer un minimum avec une moyenne par exemple).
- Multiplexage/démultiplexage [66] : similaires respectivement à un "group by" ou à une "union", et sont utilisés pour décomposer et fusionner des flux.
- Recherche d'items fréquents [34; 91] : requêtes top-k ou basées sur un seuil.
- Jointure : de plusieurs flux ou de flux et de données statiques.
- Fenêtrage : toutes les requêtes précédentes doivent être contraintes à retourner des résultats dans une fenêtre (physique ou logique).
- Fouille de flux : opérations telles que la recherche de motifs, la recherche de similarité, et la prédiction.

Comme pour les SGBD, les SGFD utilisent un langage de haut niveau pour l'expression des requêtes. Les requêtes de Stream [14] et de TelegraphCQ[32] sont formulées à l'aide d'un langage relationnel [15] proche de SQL "SQL-like" dans lequel il est possible de définir des fenêtres. Dans Aurora [1], un plan de requête est créé à partir d'une interface graphique. On place des boîtes qui représentent des opérateurs prédéfinis et des flèches qui représentent les flux de données. Les opérateurs (filtres, jointures, etc.) peuvent porter sur des flux, sur des fenêtres ou encore sur des tables statiques. Il est intéressant de disposer d'opérateurs faisant le lien entre ces types de structures.

2.6 Délestage dans les SGFD

Un des défis d'un système de gestion de flux de données est de pouvoir traiter les flux de données en temps réel malgré les limites des ressources (CPU, mémoire, bande passante,etc.). Les requêtes continues, et la volumétrie des données qui arrivent d'une façon imprévisible peuvent causer la latence du traitement. Il faut donc minimiser le temps de latence et maximiser la capacité de traitement. Deux méthodes peuvent être utilisées : la planification de l'exécution des opérateurs, ou le délestage (Load Shedding) c'est-à-dire laisser tomber une partie de données reçues. La précision de réponses est affectée par les deux méthodes. Il faut donc trouver un compromis acceptable (les bonnes techniques d'approximation). La première méthode (planification) permettant d'optimiser le plan d'exécution d'une requête, n'est pas traitée dans ce rapport, on peut trouver des détails sur la planification dans [18; 29].

Le load shedding peut se faire aléatoirement comme dans Stream, ou en analysant des données cruciales (des statistiques par exemple) comme dans Aurora. On présente dans ce qui suit deux systèmes de gestion de flux de données appliquant le Load Shedding dans leurs stratégies d'optimisation : Stream et Aurora.

2.6.1 Délestage dans Aurora

Dans [106], on définit le délestage comme une insertion automatique d'opérateurs "Drop" dans un réseau actif. Un opérateur "Drop" envoie moins de n-uplets en sortie que ce qu'il reçoit en entrée. Les auteurs considèrent deux types de délestage : aléatoire et sémantique.

Le processus de délestage consiste en trois décisions : le moment du load shedding (Quand ?), l'emplacement du load shedding dans le cycle du traitement (Où ?) et la quantité des n-uplets à éliminer (Combien ?). Pour prendre ces décisions, on exploite des informations de qualité de service (QoS pour Quality Of Service) en plus de quelques statistiques concernant le système tels que le coût de service et la sélectivité. La QoS est modélisée comme un ensemble de fonctions qui relient un paramètre de sortie à son utilité, ces fonctions permettent par exemple de déterminer l'importance des valeurs de sortie, et l'utilité selon le pourcentage des n-uplets délivrés.

Le système évalue d'une façon continue la charge du système en calculant une équation de "charge" (Load Equation) basée sur des paramètres du système et le taux d'arrivée des u-uplets. Lorsqu'un dépassement de capacité est observé, le délestage est déclenché. Les n-uplets peuvent être supprimés à tout endroit du plan d'exécution de la requête (avant tout opérateur). En positionnant les opérateurs "Drop" en début du réseau, on sauve plus de cycles de traitement. Une fois qu'on a déterminé les endroits possibles où placer les "Drops", on calcule un ratio perte/gain pour chacun des endroits. Pour minimiser la perte de précision et gagner plus de cycles de traitement, les "Drops" sont insérés à ces endroits dans un ordre croissant du ratio. Enfin, il reste à déterminer la quantité des n-uplets à éliminer. Dans le cas du délestage aléatoire, il faut décider du pourcentage de n-uplets à rejeter. Tandis que dans le cas du délestage sémantique, on doit aussi décider de la nature des n-uplets à supprimer. Cette décision est faite grâce aux fonctions QoS et les histogrammes des valeurs de sorties permettant de donner les intervalles des valeurs des n-uplets liés à leur utilité et leur fréquence relative. On commence à rejeter des n-uplets à partir des intervalles les moins "utiles". Les expérimentations ont montré que le délestage sémantique est plus performant que l'aléatoire, il cause moins de perte d'utilité et donc de précision de la réponse délivrée.

2.6.2 Délestage dans Stream

Babcock et al. [19; 20] s'intéressent aux requêtes d'agrégation, et plus particulièrement, les opérateurs de calcul de moyenne, de minimum, de médiane, appliqués aux fenêtres glissantes des éléments les plus récents. La technique proposée est d'introduire un opérateur d'échantillonnage uniforme (load shedder) à différents points du plan de la requête. Chaque load shedder est paramétré par un taux d'échantillonnage. Une opération de Load Shedding est lancée en fonction d'une équation de charge (Load equation) régie par les paramètres liés aux opérateurs.

Les limites de Hoeffding permettent de déterminer le taux d'échantillonnage effectif (produit de taux d'échantillonnage de tous les opérateurs formant une requête) pour une requête donnée. Quant au placement des load Shedders, s'il n'y a pas

d'opérateurs partagés entre les requêtes, le problème est simple, il suffit d'introduire un load shedder avant le premier opérateur de chacune des requêtes. Par contre si quelques opérateurs sont partagés entre de multiples requêtes la situation devient plus compliquée. [20] a défini un algorithme permettant d'affecter de façon optimale les taux d'échantillonnage dans les endroits possibles du "réseau" d'exécution d'une requête.

2.7 Quelques SGFD

Le nombre croissant d'applications faisant intervenir des flux de données ont motivé les propositions de plusieurs SGFD dans la communauté académique, parmi lesquels, STREAM [14], Aurora [1], Medusa [116], Borealis [2] et TelegraphCQ [32] qui ont une vocation généraliste, alors que d'autres sont destinés à un type particulier d'application (tels que Gigascope [46], OpenCQ [89], StatStream [118], NiagaraCQ [36], Xfilter [13]). En outre, les offres commerciales issues de ces recherches commencent aussi à émerger telles que Aleri [9], StreamBase [105] et Truviso [110]. Nous avons dans le cadre de cette thèse [3] choisi deux SGFD à usage public et à vocation généraliste : STREAM [14] et TelegraphCQ [32] et nous avons établi une étude expérimentale dans le but d'analyser les avantages et les limites de ces deux systèmes. Ces deux systèmes sont présentés dans les sous-sections suivantes 2.7.1 et 2.7.2. L'étude expérimentale est présentée dans notre article [3].

2.7.1 STREAM

STREAM (pour STanford stREam datA Management) est un système prototype de gestion de flux de données généraliste développé à l'Université de Stanford [14]. Ce système permet d'appliquer un grand nombre de requêtes déclaratives et continues à des données statiques (tables) et dynamiques (flux de données).
Un langage déclaratif CQL (Continuous Query Language) [15], dérivé de SQL, a été implémenté pour pouvoir traiter des requêtes continues. STREAM considère deux structures de données :

- Flux : un flux S est un ensemble d'éléments (s,t), où s est un n-uplet appartenant au flux, et t $\in$T est l'estampille temporelle (timestamp) du n-uplet, t croissant de façon monotone.
- Relation : une relation R(t) est un ensemble de n-uplets qui dépend du temps. A chaque instant une relation est susceptible d'avoir un nouveau contenu.

CQL supporte des fenêtres glissantes sur un flux S, qu'on peut décliner en 3 groupes : (1) fenêtre basée sur le temps T (S[Range T]), (2) fenêtre basée sur le nombre de n-uplets N (S[Rows N]), et (3) fenêtre partitionnée (S[Partition By $A_1, ..., A_k$ Rows N]) similaire au Group By en SQL. Le fenêtrage est considéré comme un opérateur transformant un flux en une relation ayant le même schéma que le flux.
Le langage CQL spécifie des opérations de sélection, d'agrégation, de jointure, de fenêtrage sur les flux, ainsi que trois opérateurs transformant des relations en flux :

- Istream(R(t)) : envoie sous forme de flux les nouveaux n-uplets apparus dans R à la période t ;
- Dstream(R(t)) : envoie sous forme de flux les n-uplets supprimés de R à la période t ;
- Rstream(R(t)) : envoie sous forme de flux tous les n-uplets appartenant à R à la période t.

L'utilisateur saisit ses requêtes grâce à une interface graphique ou en utilisant un fichier XML. Un plan d'exécution de la requête est créé par le système. Il est composé d'opérateurs, de buffers (pour garder temporairement les données non encore traitées) et de résumés de certaines données (pour donner des réponses approximatives en cas de surcharge du système).

2.7.2 TelegraphCQ

TelegraphCQ est un Système de Gestion de Flux de Données réalisé à l'Université de Berkeley [32]. Ce système est construit comme une extension du SGBD relationnel PostGreSQL pour supporter des requêtes continues sur des flux de données. Le format d'un flux de données est défini comme toute table PostGreSQL dans le langage de définition de données de PostGreSQL, et est créé à l'aide de la clause CREATE STREAM, avant d'être utilisé dans des requêtes continues. A chaque source de flux est affecté un adaptateur (wrapper) qui transforme les données en un format compatible avec les types de données PostGreSQL. TelegraphCQ est un mode d'exécution de PostGreSQL, l'utilisateur pouvant aussi lancer un serveur PostGreSQL en mode normal.
Les requêtes continues peuvent inclure une clause de fenêtrage sur les flux. Une fenêtre est définie sur un intervalle de temps, l'estampille temporelle (timestamp) de chaque n-uplet du flux étant assigné par l'adaptateur si celui-ci ne possède pas un attribut déclaré comme timestamp. La sémantique de fenêtrage a évolué au cours des différentes versions de TelegraphCQ. La version la plus récente définit la clause de fenêtrage par [RANGE BY 'interval' SLIDE BY 'interval' START AT 'date et heure de début'] :

- RANGE BY : définit la taille de la fenêtre en terme d'intervalle de temps ;
- SLIDE BY : définit l'intervalle après lequel la fenêtre est recalculée, spécifié en intervalle de temps ;
- START AT : définit l'instant (date et heure) auquel la première fenêtre commence, dans un format de date standard.

Les flux interrogés par les requêtes peuvent être archivés dans la base de données PostgreSQL à la demande de l'utilisateur (clause ARCHIVED). Dans le cas contraire, ils sont supprimés du système lorsque les requêtes n'ont plus à les conserver dans des espaces temporaires pour leur exécution.

2.8 Conclusion

De plus en plus de données circulent sous la forme de flux (data streams) de façon continue, à un rythme rapide (par rapport aux capacités de traitement du système qui les reçoit) et éventuellement de manière infinie. Les techniques classiques d'interrogation et d'extraction de connaissances sur des données statiques deviennent inadaptées à un contexte aussi dynamique puisque la majorité de ces approches requiert plusieurs passages sur les données. Les Systèmes de Gestion de Flux de Données ont été développés afin de traiter ces données arrivant en continu et à un taux rapide. La différence de ces systèmes avec les Systèmes de Gestion de Bases de Données résident dans le fait que les données sont persistantes dans les SGBD et les requêtes sont transitoires (éphémères). Alors que dans les SGFD, ce sont les requêtes qui sont persistantes et les données sont transitoires. Les données continues sont considérées comme des flux infinis de données sur lesquels les requêtes sont évaluées de manière continue. Pour des opérateurs bloquants, des fenêtres temporelles ou des fenêtres logiques (en nombres d'éléments) sont utilisées afin de construire un ensemble fini d'éléments à partir des flux infinis.

Les requêtes sur les flux de données concernent généralement des tâches de supervision et d'envoi d'alertes. Ces tâches peuvent nécessiter de garder une trace des données historiques. Ne pouvant pas stocker la totalité des données des flux en raison des ressources limitées, il est souvent utile voire nécessaire de disposer de techniques de résumé efficace des flux de données entrants. Ainsi, de nombreux travaux de recherche ont été récemment proposés [6] pour répondre au principal défi induit par la dynamicité des flux de données : trouver le meilleur compromis entre l'efficacité (traitement au fil de l'eau) et la précision des résultats.

Chapitre 3

Résumé de flux de données individuel

Sommaire

3.1 Introduction

La volumétrie et le taux d'arrivée des flux de données représentent des contraintes temporelles (temps de traitement) et spatiales (espace de stockage) qui doivent être prises en compte dans le processus de traitement des données. Dans plusieurs applications, des structures de résumés (appelés aussi synopsis dans la littérature) sont construites afin de répondre approximativement à des tâches de traitement de données. Csernel [47] définit un résumé sur flux de données selon deux vues : une vue fonctionnelle du résumé ou avec une vue plus large. Dans la vue fonctionnelle, un résumé vise à rassembler une partie des informations contenues dans le flux sur une période donnée de façon à pouvoir résoudre une problématique particulière sur cette période. Dans la vue plus générale, le résumé vise à garder des informations compactes sur un flux de données dans son intégralité temporelle, sans envisager de problématique a priori. Cette vue du résumé est donc bien plus contraignante, elle est aussi plus difficile à évaluer, car par nature, un bon résumé généraliste doit pouvoir permettre n'importe quel type d'analyse et fournir des résultats satisfaisants à défaut d'être précis. C'est cette seconde vue qui nous intéresse dans le cadre de la problématique de la thèse. En effet, l'objectif de notre étude est de maintenir un

historique des flux de données provenant de multiples capteurs de telle sorte qu'on puisse répondre à des traitements sur le présent et sur le passé des flux telles que : l'agrégation des flux, l'estimation de la fréquence, l'estimation des quantiles, détection de changement, etc.

Il existe une variété de techniques qui peuvent être utilisées pour la construction des résumés de flux de données. Nous détaillons dans ce qui suit quelques unes de ces méthodes qui ne sont que des améliorations de techniques qui existaient déjà dans le cadre du traitement des données statiques pour des données massives. La plupart de ces méthodes respectent les contraintes liées à notre problématique, à savoir :

- **La généricité :** les applications nécessitant des structures de résumés sont nombreuses, il est donc préférable de construire un résumé qui soit assez générique. Ainsi le temps de traitement et l'espace de stockage sont optimisés, puisqu'on n'a pas besoin de construire un résumé pour tout type d'application.
- **Une seule passe** : en raison de la volumétrie des données provenant des flux, on a besoin d'un algorithme permettant de faire une seule passe sur les données et générant une réponse rapide.
- **Des ressources limitées** : il s'agit des contraintes en temps de traitement et en espace de stockage qui doivent être optimisées. Ces contraintes sont assez classiques dans l'élaboration d'un résumé et plus généralement dans le traitement de données massives.
- **La dynamicité** : le résumé doit s'adapter aux évolutions rapides des données du flux. Les résumés peuvent être utilisés pour répondre à des requêtes de prévisions et doivent donc être sensibles à d'éventuels changements dans les données.

Nous avons limité notre étude aux techniques qui s'appliquent à des données quantitatives et uni-dimensionnelles. Néanmoins, certaines des techniques présentées dans ce chapitre peuvent aussi être appliquées à des données qualitatives et/ou multi-dimensionnelles. Nous avons classé les techniques de résumé selon deux catégories. La première catégorie concerne les *techniques probabilistes* où le résumé a une probabilité donnée d'être sélectionné. Ceci signifie que si on applique ces techniques à plusieurs reprises sur le même jeu de données, le résultat à l'issue du résumé pourra varier. (2) La seconde catégorie concerne les *techniques déterministes*. L'application répétitive de cette dernière catégorie sur le même jeu de données fournira toujours le même résultat.

3.2 Techniques déterministes

3.2.1 Histogramme

L'histogramme est un outil fréquemment utilisé, pour résumer des données discrètes ou continues. Il est employé pour montrer les caractéristiques principales de la distribution des données de façon pratique. Un histogramme sépare les valeurs possibles des données en classes ou groupes. Pour chaque groupe, un rectangle est construit, sa base correspond aux valeurs de ce groupe, et sa surface est propor-

tionnelle au nombre d'observations dans le groupe. Cela signifie que les rectangles sont de hauteur différente. L'espace de stockage est représenté par le nombre de segments dans l'histogramme. Il est relativement facile d'utiliser l'histogramme pour répondre à différentes sortes de requêtes telles que les requêtes sur les intervalles. En effet, il suffit de déterminer l'ensemble des segments qui se trouvent au-delà du seuil spécifié par l'utilisateur. Un certain nombre de stratégies peuvent être élaborées pour améliorer la résolution des requêtes à partir de l'histogramme [98; 99]. Il existe plusieurs types d'histogrammes proposés dans la littérature. Les plus populaires sont :

- Histogramme V-optimal (ou équivariant) : permet d'approcher la distribution d'un ensemble de valeurs $v_1, v_2, ..., v_n$ par une fonction en escalier $\hat{v_i}$ de telle sorte qu'on minimise la somme des erreurs quadratiques $\sum_i (v_i - \hat{v}(i))^2$. Généralement on utilise la variance comme critère d'optimisation. Cette approche produit des segments homogènes en terme de variance.
- Histogrammes à profondeur uniforme (ou équidistribué) : les frontières des cases de l'histogramme sont choisies de façon à obtenir des segments à effectifs égaux.
- Histogrammes à fin biaisée : sont similaires aux histogrammes à profondeur uniforme si ce n'est qu'ils n'enregistrent les valeurs qu'au dessus d'un certain seuil. Ils permettent ainsi de répondre à des requêtes de type 'iceberg' (agrégats au-delà d'un seuil spécifié) [59].

3.2.2 Compression par ondelettes

L'idée majeure de la compression par ondelettes est de décomposer un ensemble de données en entrée, produisant ainsi une série de coefficients d'ondelettes et de ne conserver qu'un sous ensemble de ces derniers. Ceci peut se concevoir comme une tâche d'approximation ou de compression. Cette procédure se fait avec perte et introduit donc une erreur lors de la reconstruction des données originales. L'essentiel des travaux récents se concentre sur la définition des paramètres utiles qui rendent cette erreur de reconstruction minimale tout en prenant en compte la contrainte d'espace limité.

L'utilisation des ondelettes dans un cadre de flux de données permet de faire face à la volumétrie des données. Malheureusement, les algorithmes de construction de résumés par ondelettes ont été conçus pour des données statiques (stockées sur disque) et effectuent plusieurs passes sur les données, ce qui n'est pas envisageable dans le cas des flux de données. Nous décrivons dans ce qui suit la méthode de compression par ondelettes classique puis nous présentons les travaux effectués dans le but d'adapter la compression aux flux de données.

Compression par ondelettes

La transformée par Ondelettes [103](comme la transformée de Fourier) est un outil mathématique permettant de saisir l'évolution de fonctions numériques. Souvent, il peut être suffisant de garder très peu de coefficients d'ondelettes pour saisir

les tendances de fonctions numériques. Bien que la théorie des ondelettes soit vaste, nous présenterons brièvement le processus de décomposition en ondelettes en appliquant le cas particulier des ondelettes de Haar sur un exemple simple. L'idée de base des techniques d'ondelettes est de créer une décomposition des caractéristiques des données en un ensemble de fonctions d'ondelettes et fonctions de base. La propriété de la méthode d'ondelette est que les coefficients d'ordre supérieur de la décomposition illustrent les grandes tendances des données, tandis que les tendances les plus localisées sont capturées par les coefficients d'ordre inférieur.

On considère une série de données

$$a = [2, 2, 0, 2, 3, 5, 4, 4]$$

de taille $N = 8$. Sa décomposition en ondelettes de Haar est donnée par le tableau 3.1. On représente a de manière différente en exploitant une éventuelle corrélation de valeurs voisines. Pour ce faire, on calcule la moyenne des paires de valeurs voisines et on obtient :

$$[\frac{2+2}{2}, \frac{0+2}{2}, \frac{3+5}{2}, \frac{4+4}{2}] = [2, 1, 4, 4]$$

Cette représentation est avec perte. Afin de représenter la série initiale a, on enregistre également les coefficients dits de détail qui représentent la perte d'information. Dans la décomposition de Haar, ces coefficients sont les différences entre les moyennes et les valeurs initiales. Dans notre exemple, le coefficient de détail de la première paire des valeurs est $2 - 2 = 0$, pour la seconde paire, il est de $1 - 2 = -1$, etc. On répète cette procédure sur les moyennes obtenues jusqu'à n'avoir qu'une seule moyenne. La transformée de a est donnée par

$$w_a = [\frac{11}{4}, \frac{-5}{4}, \frac{1}{2}, 0, 0, -1, -1, 0]$$

Chaque entrée de w_a, que ce soit un détail ou une moyenne, est appelée coefficient d'ondelettes.

Resolution	Moyennes	Coefficients de détail
3	$[\frac{11}{4}, \frac{-5}{4}, \frac{1}{2}, 0, 0, -1, -1, 0]$	————
2	$[2, 1, 4, 4]$	$[0, -1, -1, 0]$
1	$[\frac{3}{2}, 4]$	$[\frac{1}{2}, 0]$
0	$[\frac{11}{4}]$	$[\frac{-5}{4}]$

TABLE 3.1 – Décomposition en ondelette de l'exemple a

Afin de mieux appréhender la nature hiérarchique de la décomposition de Haar, on utilise une structure arborescente dite *arbre d'erreur*. L'arbre correspondant à notre exemple est représenté dans la figure 3.1. Chaque nœud interne c_i correspond à un coefficient ondelette, le nœud racine correspond à la moyenne générale. Les nœuds feuilles $a[i]$ correspondent à chaque entrée de la série originale. Nous remarquons à partir de cette représentation que la reconstruction de tout $a[i]$ dépend de $\log(N+1)$

coefficients dans le chemin entre la racine C_0 est $a[i]$. Par exemple, on reconstruit $a[5]$ en additionnant (branche gauche) ou en soustrayant (branche droite) les coefficients à partir de la racine, ce qui donne : $a[5] = c_0 - c_1 + c_3 - c_6 = 11/4 - (-5/6) + 0 - (-1) = 5$. De même, on peut noter que la valeur d'un coefficient d'ondelette ne dépend que d'un sous-ensemble des valeurs d'origine, en fonction de la hauteur de l'arbre auquel ils appartiennent. Par exemple, la valeur du coefficient c_5 ne dépend que des valeurs de $a[2]$ et $a[3]$.

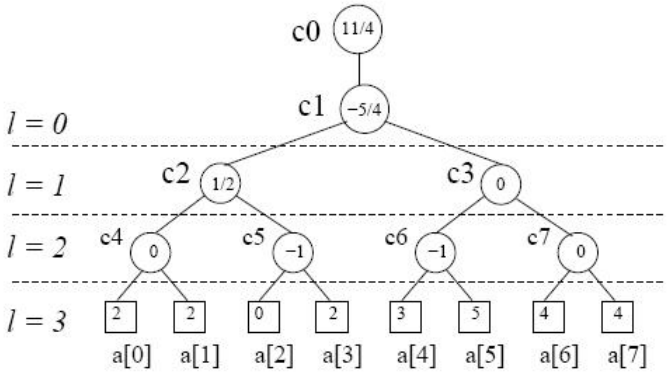

FIGURE 3.1 – l'arbre des erreurs de l'exemple a

Intuitivement, les coefficients d'ondelettes qui sont proches de la racine de l'arbre ont un poids élevé quant à la reconstruction de la série d'origine. Afin de donner la même importance à tous les coefficients, les valeurs de ces derniers sont normalisées par un facteur de $\sqrt{\frac{N}{2^l}}$ où l est le niveau du coefficient dans l'arbre d'erreur. Cette normalisation a deux conséquences importantes [103] :

1. L'énergie de la série a (sa norme L_2) est préservée. Autrement dit, grâce au théorème de Parseval, on démontre que $\|a\|_2^2 = \sum_i a[i]^2 = \sum_i (c_i^*)^2$ où c_i^* sont les valeurs normalisées des coefficients d'ondelette.
2. Le maintien des B coefficients d'ondelette les plus grands en terme de valeur absolue normalisée donne un résumé optimal en terme de somme d'erreur quadratique (SSE pour Sum-Squarred-Error).

Plus formellement, soit $\Lambda \subset w_a$ un résumé par ondelette et $\hat{a}$ la série reconstruite à partir de ce résumé, SSE est défini par $\sum_i (a[i] - \hat{a}[i])^2 = \sum_{\forall c_j \notin \Lambda} (c_j^*)^2$. Cette équation est due au théorème de Parseval. En d'autres termes, SSE est égale à la somme des carrés des valeurs normalisées des coefficients non stockés. Toutefois, l'utilisation de SSE a un principal inconvénient qui est le fait d'ignorer le comportement des séries localement dans le processus d'optimisation. Par conséquent, les erreurs de reconstruction peuvent être très accentuées pour quelques parties de la série. Dans de nombreux cas, la métrique L_∞ (erreur maximale) offre plus de garanties puisque les erreurs sont réparties sur les différents coefficients de manière plus égale.

Ondelettes et flux de données

Un environnement de flux de données introduit des contraintes supplémentaires aux algorithmes de traitement des données, à cause de la volumétrie et du taux d'arrivée des éléments. En raison des contraintes de stockage et de performance, les données doivent être traitées à la volée. Les algorithmes de flux en ligne sont restreints à ne faire qu'un passage sur les données. Dans le contexte de cette section, l'objectif est de construire et de maintenir à jour un résumé par ondelette optimal d'une série dont les données arrivent sous forme de flux. Il existe plusieurs façons de modéliser un flux [68]. Nous allons étudier la problématique des ondelettes pour les deux modèles suivants : (1) Modèle de série temporelle, où des nouveaux éléments sont insérés dans la série, c'est à dire que le i^{eme} valeur est $a[i]$, c'est le cas par exemple du flux fourni par un capteur de température correspondant à la température mesurée à chaque instant. (2) Modèle du tourniquet, où les éléments du flux mettent à jour les données de la série a. C'est-à-dire que chaque élément du flux (i, u) est une mise-à-jour de l'ancienne valeur de $a[i]$ qui deviendra : $a[i] + u$.

Modèle de série temporelle Les éléments sont ajoutés à la fin de la série dans ce modèle. Seuls les coefficients appartenant au chemin de la racine aux nouveaux éléments vont être modifiés. Cci signifie que la majeure partie des coefficients d'ondelettes, excepté ceux appartenant au sous-ensemble de taille logarithmique (C.f. section précédente) ne seront pas affectés par l'arrivée des nouveaux éléments. Cette observation implique un algorithme très simple pour construire un résumé en B coefficients en utilisant une métrique L_2 : on maintient les B coefficients les plus élevés en terme de valeurs normalisées absolues parmi les valeurs finalisées, ainsi que les coefficients mis-à-jour ($\log(N)$ coefficients). Ces derniers sont ensuite comparés avec les B coefficients pour construire un nouveau sous-ensemble de coefficients de valeurs élevées. Tel qu'il est indiqué dans [83], cette méthode peut mal estimer une partie du flux de données. Garofalakis et Kumar [64] proposent un algorithme déterministe basé sur la programmation dynamique, qui donne une bonne approximation en B coefficients en minimisant le maximum des erreurs relatives ou le maximum des erreurs absolues. Toutefois, cette solution optimale a une complexité élevée en temps et en mémoire ce qui la rend inapplicable à des données volumineuses comme c'est le cas dans les flux de données. [83] propose des méthodes de construction de résumé par ondelettes en une seule passe en minimisant le maximum des erreurs (relative et absolue). Les auteurs utilisent une heuristique permettant de réduire la complexité de l'algorithme de décomposition et tout en s'approchant des performances de [64] en terme de minimisation des erreurs.

Modèle de tourniquet La décomposition en ondelettes des flux de données ayant un modèle de tourniquet est plus générale que le cas traité précédemment dans la mesure où tous les coefficients peuvent être affectés par l'arrivée d'un nouvel élément. En effet, ce dernier peut être associé à un élément de la série de façon arbitraire. Ceci rend plus difficile le maintien des coefficients, et par conséquent la construction du résumé. Le travail effectué dans [68] utilise un sketch afin de maintenir de façon

probabiliste un résumé incrémental du flux de données, les B coefficients d'ondelettes sont ensuite calculés à partir du sketch. [42] utilisent une méthode similaire mais au lieu de décomposer le sketch en ondelettes, les auteurs proposent de faire l'inverse, c'est-à-dire de construire un sketch à partir de la décomposition en ondelettes du flux qui sera mis à jour de façon incrémentale. Leur algorithme a le mérite d'être applicable à des flux de données à domaines très larges. De plus, il peut être étendu à des flux de données multi-dimensionnelles.

3.2.3 Segmentation de courbe

Généralités

Dans un objectif de compression, la technique de segmentation peut être utilisée afin de ne conserver que les informations les plus pertinentes sur les courbes. Il s'agit d'un moyen de découper une série temporelle en épisodes et d'associer une information à chacun de ces épisodes, le nombre d'épisodes étant fixé à l'avance. Les épisodes peuvent être de durées différentes et la courbe segmentée est représentée par une fonction en « escalier ». Cette méthode est utile d'une part pour décrire et résumer l'information, et d'autre part pour réduire le bruit en lissant la courbe. La figure 3.2 donne un exemple d'une courbe de données de consommation électrique et 10 épisodes (en pointillés) générés par segmentation.

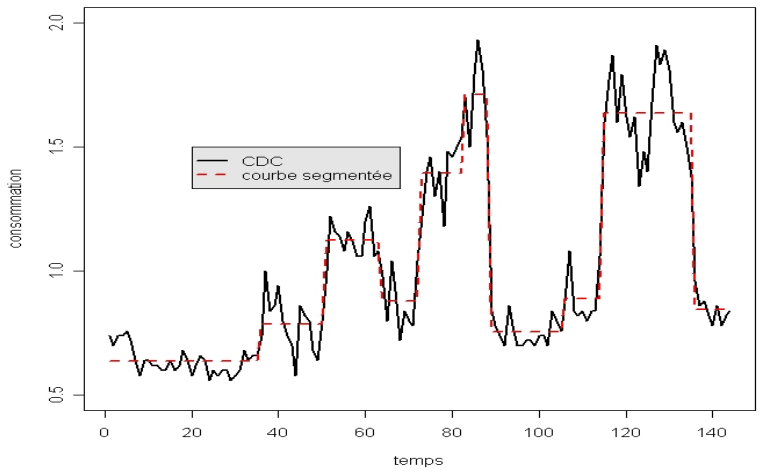

FIGURE 3.2 – Une courbe et sa segmentation en 10 épisodes

L'approche la plus simple est de segmenter une série temporelle en segments de même taille, celle-ci est le seul paramètre à prendre en compte. Cette approche est utile quand il s'agit d'une série temporelle avec un comportement régulier. Malgré la simplicité et la rapidité de cette technique, qui la rend exploitable dans un contexte de flux de données, elle ne permet pas de représenter correctement la plupart des séries temporelles issues d'applications réelles. Keogh [86] a démontré, qu'à quantité

d'information égale, la segmentation avec taille égale s'accompagne d'une perte d'information plus importante par rapport à d'autres techniques de segmentation (sans contrainte de taille).

Soit une courbe C de p points, notés $c_1, c_2, ..., c_p$. Pour un découpage en k segments, on note $c_{1l}, ..., c_{il}, ..., c_{n_l l}$ la séquence appartenant au segment l avec n_l le nombre de points du segment l. Pour tout c_{il}, on note $\hat{c}_l$ le représentant du segment l. La courbe segmentée est alors représentée par $\hat{C} = ((\hat{c}_1, n_1), (\hat{c}_2, n_2), ..., (\hat{c}_k, n_k))$ où chaque épisode l est défini par $\hat{c}_l$ qui est le représentant numérique des n_l points de la courbe initiale C.

La valeur numérique $\hat{c}_l$ associée à un segment doit représenter l'ensemble des valeurs des points pris au cours du segment en question. La segmentation d'une courbe est donc la recherche de l'ensemble des segments et des valeurs numériques associées à ceux-ci de façon à optimiser un critère donné. Généralement, le critère utilisé est la somme des erreurs quadratiques calculée par l'expression suivante :

$$\sum_{l=1}^{k} \sum_{i=1}^{n_l} (c_{il} - \hat{c}_l)^2$$

Dans ce cas, le représentant numérique est la moyenne des points de la courbe initiale qui définissent un segment.

On pourrait énumérer l'ensemble des solutions et choisir celle qui optimise le critère de recherche (minimiser la somme des erreurs quadratiques), ceci est d'un coût prohibitif en temps de calcul. Plusieurs stratégies [78] ont été proposées dans la littérature. On peut les distinguer en trois types selon leur rapidité et leur capacité à trouver la solution optimale : stratégie agrégative ascendante (développée par Jain, Murty et Flynn en 1999 et appliquée aux données massives), stratégie hiérarchique descendante (développée par Duda et Hart en 1979, s'appuie sur des divisions successives des segments) et la stratégie optimale que nous allons développer dans la sous-section suivante puisqu'elle fait partie des techniques de résumé utilisées dans le cadre de cette thèse.

Segmentation optimale

Le problème de la segmentation d'une série temporelle a une solution exacte grâce à la programmation dynamique [25] avec une complexité en $O(kp^2)$, où p est le nombre de points de la série à segmenter et k le nombre de segments. Voici l'algorithme de calcul de la k-segmentation optimale d'une courbe C de p éléments, avec les notations suivantes :

- $seg(C[i, j])$: la segmentation optimale en un seul segment de la série extraite de la courbe C entre les indices i et j.
- $seg^l_{i,j}$ est la segmentation obtenue à partir de la segmentation optimale en un seul segment de i à j et de la segmentation optimale en l segments de j à p.
- $segOpt^l_i$ est la segmentation optimale en l segments de série extraite entre i et p.

Algorithm 1 Algorithme segmentationOptimale

Require: k - Nombre de segments
Require: C - Courbe de la période t (p élements)
Ensure: $segOpt_1^k$
  for i de p à 1 **do**
    $segOpt_i^1 \leftarrow seg(C[i:p])$
  end for
  for j from 2 to k **do**
    for i from 1 to $p-j+1$ **do**
      for l from i to $p-j+1$ **do**
        $seg_{i,l}^k \leftarrow (seg(C[i,l]), segOpt_{l+1}^{k-1})$
      end for
      $segOpt_i^j \leftarrow seg_{i,l_{opt}}^j$
      tel que $sse(seg_{i,l_{opt}}^j) \leq sse(seg_{i,l}^j) \quad \forall l \in \{i, ...p-k+1\}$
    end for
  end for
  return $segOpt_1^k$

L'algorithme (1) fonctionne en décomposant la segmentation optimale en k segments en une segmentation en 1 segment et une segmentation optimale en $k-1$ segments. On énumère ces segmentations et on choisit celle dont le SSE (somme des erreurs quadratiques) est le plus petit pour trouver la segmentation optimale.

La segmentation n'est pas vraiment une technique de résumé pour les flux de données à cause du nombre de passes effectué sur les données. Nous l'avons néanmoins citée dans ce chapitre car nous l'avons utilisée comme méthode de résumé d'un flux individuel pour valider nos approches. Nous avons décrit son adaptation à notre problématique dans le chapitre 6.

3.3 Techniques probabilistes

3.3.1 Sketch

Ce sont Alon Matias et Szegedy [12] qui ont introduit la notion de sketch aléatoire pour la première fois. Il s'agit d'un résumé de flux de données très compact utilisé afin d'estimer la réponse à certaines requêtes portant sur l'ensemble des données du flux. Cette technique vise à projeter aléatoirement chaque élément dans un espace particulier en utilisant des fonctions de hachage et à ne garder que les composantes les plus pertinentes, sauvant ainsi de l'espace tout en préservant l'essentiel de l'information. Nous présentons dans ce chapitre quelques types de sketches qui peuvent être appliqués à des flux de données.

Sketch de Flajolet Martin

Le sketch de Flajolet-Martin [62](*FM sketch*) permet d'estimer le nombre d'éléments distincts dans un flux de données. Il a été présenté en 1985 pas en tant que sketch pour les flux de données mais comme un moyen de résumer des données massives. Toutefois, sa définition est très proche de la définition d'un sketch et a le mérite d'être incrémental et rapide, ce qui le rend adaptable pour les flux de données. Ses performances se dégradent avec le nombre d'éléments différents insérés. Néanmoins, les bornes permettent de fixer les paramètres en fonction du nombre d'occurrences attendues. Son fonctionnement est le suivant : soient un tableau binaire de taille L initialisé à 0 et une fonction de hachage $h(x)$ qui répartit uniformément les éléments d'un ensemble M dans l'ensemble des mots binaires de taille L. Soit $pfp(h(x))$ la position de 1 de plus faible poids dans la représentation binaire de x. Le bit d'indice $pfp(h(x))$ est alors mis à 1 dans le sketch s'il était encore à 0. Une fois le sketch construit, on estime le nombre de valeurs distinctes dans M par p la position du 0 de plus faible poids dans le tableau binaire. On peut prouver que $E(p) = log_2(\phi n)$ avec $\phi = 0.77351$, $\sigma(r) = 1.12$ et n le nombre de valeurs distinctes dans le flux. Pour plus de détails sur l'obtention de ces bornes, le lecteur peut se référer à [62].

Count sketch

Le sketch proposé par [34] permet de répondre aux requêtes de type '*top-k*', c'est à dire trouver la liste de k éléments les plus fréquents dans le flux de données respectant la contrainte $n_i > (1 - \epsilon)n_k$ où n_i est la fréquence d'un élément donné d'indice i dans la liste des éléments classés par fréquence et ϵ la précision souhaitée. Ce sketch suppose que le flux de données est sous forme de séquence d'éléments tirée d'un espace de valeurs M. Le sketch est représenté par une matrice de comptage CS de taille $t \times b$. On dispose de t fonctions de hachage s_i de M dans $\{-1, 1\}$ et de t fonctions de hachage h_i de M dans $[1, b]$. A l'arrivée d'un nouvel élément e, on met à jour la matrice par :

$$\forall i = 1, ..., t \quad CS(i, h_i(e)) += s_i(e)$$

On estime la fréquence d'un élément donné e en évaluant la médiane de $h_i(e)s_i(e)$. Enfin, pour obtenir les 'top-k' éléments, on trie incrémentalement les fréquences des éléments et on garde dans une liste celles qui respectent la contrainte. Ainsi, chaque nouvel élément est inséré dans le sketch, s'il se trouve par ailleurs dans la liste, sa fréquence dans la liste est mise à jour et la liste est triée. Si il n'est pas dans la liste, sa fréquence est estimée à l'aide du sketch, si cette dernière est plus grande que celle de l'élément de plus petite fréquence présent dans la liste, on l'ajoute et on enlève l'autre.

Filtre de Bloom

Le filtre de Bloom [26; 37] est un tableau de bits qui permet de tester d'une manière rapide l'appartenance d'un élément à un certain ensemble d'éléments. Le

filtre de Bloom consiste en deux composants : un ensemble de k fonctions de hachage et un vecteur de bits d'une taille L donnée. Toutes les fonctions de hachage sont configurées de telle sorte que leurs intervalles correspondent à la taille du vecteur. Par exemple, si le vecteur de bits est de taille 200, alors toutes les fonctions de hachage doivent retourner une adresse entre 1 et 200. Les fonctions de hachage garantissent que les adresses générées sont réparties de façon équiprobable sur toutes les valeurs possibles. Pour introduire un élément dans un filtre de Bloom, on calcule les valeurs des fonctions de hachage et on active (on met à 1) les bits du vecteur correspondants. Pour tester l'appartenance d'un élément x à un ensemble M d'éléments introduits dans le filtre de Bloom, on lui applique les fonctions de hachage. Si au moins un des bits est à 0 alors x n'appartient pas à M. Par contre, si tous les bits sont à 1 alors x appartient probablement à M avec un taux de faux positif moyen donné par la relation suivante :

$$f = (1 - e^{-kM/L})^k$$

où L est la taille du vecteur du filtre de Bloom, M le nombre d'éléments ajoutés au filtre et k le nombre de fonctions de hachage. Pour minimiser ce taux, on choisit un nombre de fonctions de hachage k respectant la relation suivante :

$$k = ln2(L/M)$$

Un certain nombre d'améliorations ont été développées autour des Filtres de Bloom afin de palier aux limites de ces résumés. Le *Counting Bloom Filter* [58] permet de gérer la suppression d'un élément du sketch. Ce qui le rend applicable aux fenêtres glissantes où des éléments ne faisant pas partie de la fenêtre sont considérés comme expirés et ne doivent plus faire partie du sketch. Les *Scalable Bloom Filters* [10] permettent de construire des filtres de Bloom dont on peut augmenter la taille de façon dynamique en garantissant un taux d'erreur maximal fixe.

Count min sketch

Le COUNT MIN sketch [43] est conçu pour les flux de données dont les éléments correspondent à une mise à jour positive d'un attribut donné. Soit $a(t) = a_1(t), ..., a_n(t)$ l'état courant des attributs a_i à l'instant t. La mise à jour de ce vecteur se fait à l'arrivée d'un couple (i_t, c_t) par :

$$a_i(t) = a_i(t-1) + c_t \text{ et } a_j(t) = a_j(t-1) \text{ pour } j \neq i$$

La structure de ce sketch est définie par deux paramètres (ϵ, δ) qui sont respectivement la précision et la probabilité d'échec. On pose $w = \lceil \frac{e}{\epsilon} \rceil$ et $d = \lceil \frac{1}{\delta} \rceil$. Le sketch est stocké dans une matrice COUNT de taille $d \times w$ initialisée à 0. d est le nombre de fonctions de hachage aléatoires uniformes deux à deux indépendantes de $[1, d]$ dans $[1, w]$. Pour tout instant t, on met à jour le tableau COUNT ainsi :

$$\forall j \in 1, ..., d \quad \text{COUNT}[j, h_j(i_t)] = \text{COUNT}[j, h_j(i_t)] + c_t$$

A l'instant t, on obtient pour chaque identifiant $i \in 1, 2, ..., n$ un estimateur $\hat{a}_i(t)$ du compte $a_i(t)$ par :

$$\hat{a}_i(t) = \min_j \text{COUNT}[j, h_j(i)](t)$$

Ainsi, cette méthode de sketch permet de répondre à la requête de type $Q_0(i) = a_i(t)$ mais aussi à $Q_1(j,k) = \sum_{i=j}^{k} a_i(t)$ et $Q_2(a,b) = \sum_{i=1}^{d} a_i(t)b_i(t)$.

3.3.2 Approche par clustering

Tout comme en classification automatique classique, la classification de flux de données vise à catégoriser les éléments et les regrouper dans des classes en se basant sur des données statistiques. Plusieurs méthodes ont été proposées par la communauté de fouille de données, mais la plupart d'entre elles ne peuvent pas s'appliquer à des flux de données. Les algorithmes de classification de flux de données doivent répondre à toutes les contraintes d'ordre générale sur les algorithmes de traitements de flux de données, et doivent donc travailler en une seule passe et avec une utilisation des ressources mémoires par élément traité faible. Les analyses doivent aussi porter sur l'intégralité des éléments observés ou sur une partie de ceux ci définie a posteriori. De plus, les flux de données nécessitent une étape *en ligne*, pendant laquelle les données sont traitées de façon continue, et une étape *hors ligne* basée sur les besoins d'analyse. Afin de répondre à ce besoin, des méthodes de micro-clustering ont été proposées comme étape préliminaire de compression du flux qui préserve les informations temporelles sur les données, ensuite vient l'étape de classification du flux de données à proprement parler. Nous présentons quelques méthodes de clustering adaptées aux flux de données puis citons deux techniques utilisant le micro-clustering afin de résumer les données des flux.

Clustering

Guha et al. [72; 73] ont étudié l'adaptation de la technique du k-median au contexte des flux de données. L'algorithme que proposent les auteurs permet d'effectuer une seule passe sur les éléments tout en utilisant un espace mémoire réduit. Il requiert $O(nk)$ en temps et $O(n\epsilon)$ en espace où k est le nombre de centres (de classes), n le nombre d'éléments et $\epsilon < 1$. Leur approche s'effectue de façon incrémentale (divide-and-conquer) pour maintenir les k clusters. Une amélioration a été proposée par Babcock et al. [22] en utilisant un histogramme exponentiel comme structure de données afin de maintenir k clusters sur des fenêtres glissantes de N éléments(SWKM pour Sliding Window k-median) plutôt que sur l'ensemble du flux depuis son début.

Ordonez [96] a proposé plusieurs améliorations à l'algorithme de k-means pour le clustering d'un flux de données binaires. Les données sont présentées sour forme de vecteurs binaires de tailles différentes et appelés transaction. Ordonez a développé un algorithme de k-means incrémental et a démontré expérimentalement son efficacité sur des données réelles et des données synthétiques. L'algorithme proposé opère en une seule passe et avec une complexité de $O(Tkn)$ où T est la taille moyenne d'une transaction, n est le nombre de transactions et k est le nombre de clusters. L'utilisation des données binaires simplifie la manipulation des données qualitatives et élimine le besoin de normaliser les données. L'idée principale de l'algorithme proposé

est de mettre à jour les clusters après examen d'un bloc de transactions (après $\frac{n}{\sqrt{n}}$ transactions) au lieu d'une mise à jour à chaque nouvelle transaction.

O'Callaghan et al. [28] ont proposé STREAM et LOCALSEARCH, deux algorithmes de clustering de flux de données. Dans cet algorithme, on suppose que les données arrivent sous forme de paquets. Dans chaque paquet, LOCALSEARCH est réalisé et des centres de classes résultant sont conservés. LOCALSEARCH est un algorithme de classification très proche de k-means. Quand une quantité suffisante de paquets de points a été observée, les centres résultants sont à leur tour classifiés par LOCALSEARCH pour former de nouveaux centres d'un plus haut niveau. Le processus est ainsi répété à plusieurs niveaux d'hiérarchie. Les classes finales sont ensuite réalisées sur l'ensemble des centres conservés. Cette méthode a un défaut majeur que des algorithmes plus récents essaient de corriger. En effet, elle n'est pas adaptée aux traitements faisant intervenir les données récentes.

Birch

Birch [117] (*Balanced Iterative Reducing and Clustering using Hierarchies*) est un algorithme adapté à des données volumineuses. L'approche de Birch est basée sur une classification effectuée sur un résumé compact des données au lieu des données originales. C'est pourquoi il peut traiter un grand volume de données en utilisant une mémoire limitée. Cette méthode a le mérite d'être incrémentale et n'effectue qu'une seule passe sur les données. Les informations nécessaires au clustering sont organisées sous forme d'arbre équilibré avec une taille limitée. Chaque entrée de l'arbre décrit une micro-classe, et les nouveaux nœuds sont insérés sous l'entrée la plus proche. Birch utilise la notion Cluster Feature Vector (CFV en abréviation) qui est utilisée pour représenter une classe ou une sous-classe. Son principe est simple, pour un ensemble d'éléments à résumer, on conserve l'effectif de l'ensemble et, pour chaque variable, la somme de toutes les valeurs prises par cette variable dans l'ensemble des éléments ainsi que la somme de leurs carrés. A partir de ces valeurs, on peut reconstituer pour toute variable les moments d'ordre 1 et 2 facilement et fournir des estimations sur certains quantiles. Le clustering dans Birch est réalisé suivant 2 étapes principales :

1. Résumer les données afin de réduire leur taille en construisant un arbre des CFVs dont les feuilles sont des résumés de données sous forme de micro-classes.
2. Faire la classification de ces résumés en utilisant un algorithme de classification simple (e.g K-means).

CluStream

Aggarwal [7] a adapté les CFV (*Cluster Feature Vector*) pour résumer des ensembles d'éléments issus de flux de données. La plupart des algorithmes présentés jusque là permettaient de traiter des flux de données avec un nombre réduit de passes sur les données et de façon incrémentale. Cependant, ces algorithmes ne prennent pas en compte l'évolution des données et par conséquent, la qualité des classes produites devient mauvaise si les données évoluent de façon importante dans le temps.

CluStream est un algorithme de résumé généraliste [47] d'un flux de données au sens où il est capable de donner une réponse approchée pour n'importe quelle analyse sur les données originales et à n'importe quelle période de temps. Le clustering par CluStream se fait en deux étapes. Une première étape dite *en ligne* stocke périodiquement des statistiques de résumés détaillés (micro-classes). Cette étape consiste à utiliser les CFV comme résumé de chaque classe des données. Aggarwal a adapté les CFV en leur intégrant l'étiquette de temps des données comme une variable additionnelle. Ainsi, le CFV conserve aussi la somme de toutes les étiquettes de temps (mises sous une forme numérique) ainsi que la somme de leurs carrés. A son arrivée, un nouvel élément est affecté à la micro-classe la plus proche dont le CFV est mis-à-jour selon un critère donné. Si aucune micro-classe proche n'est trouvée, on crée une nouvelle avec pour seul élément le nouvel élément. Comme la taille mémoire est fixe, il est nécessaire de fusionner deux micro-clusters ou de faire disparaître une 'vieille' qui n'a plus été mise-à-jour depuis longtemps. Un critère de vieillesse est donné par Aggarwal, basé sur l'hypothèse que les éléments de temps des éléments au sein de chaque classe suivent une distribution gaussienne. Conjointement, un système de clichés est mis en place afin de stocker à intervalles réguliers l'état du système en sauvegardant l'ensemble des CFV des micro-classes. Ces clichés sont ensuite conservés selon une structure géométrique ou pyramidale afin de privilégier les périodes proches du temps courant par rapport aux périodes les plus anciennes.

Une seconde étape dite *hors ligne* utilise les informations statistiques (CFV) à des fins d'analyse. Le reconstruction d'une partie du flux de donnée se fait grâce aux clichés conservés. En effet, les propriétés mathématiques des CFV permettent d'effectuer des opérations de soustraction sur les clichés pour obtenir le contenu du flux entre deux instants et observer ainsi l'évolution des micro-classes. Ensuite, vient l'application d'algorithmes de fouilles de données classique aux micro clusters ainsi constitués.

En utilisant une approche de microclustering conjointement à un modèle de clichés, CluStream permet de capturer l'évolution du flux de données pour n'importe quel horizon temporel. Néanmoins, CluStream trouve ses limites quant au traitement des flux de très haut débit. En effet, l'ajout de chaque élément nécessite des calculs de distance avec chaque micro-classe. Une seconde limitation concerne le nombre élevé de paramètres et dont certains dépendent de la nature du flux de données et de la vitesse d'arrivée des éléments (tel que le critère de vieillesse d'une micro-classe).

3.3.3 Echantillonnage

L'échantillonnage dans les flux de données s'appuie sur les techniques d'échantillonnage traditionnelles, mais requiert des innovations significatives pour parer au problème de la longueur infinie des flux. En effet, les techniques de fenêtrage sont utilisées pour s'adapter à la nature illimitée des données. Ces fenêtres peuvent être de deux types : point de repère ou glissantes (voir la section 2.3).

Échantillonnage dans une fenêtre point de repère

Echantillonnage de type réservoir Un algorithme classique en ligne proposé par Vitter en 1985 est largement utilisé dans le cadre des flux de données : l'échantillonnage réservoir [111]. Il permet d'obtenir de façon incrémentale un échantillon uniforme aléatoire de taille fixe, sans nécessiter la connaissance de la taille du flux complet de données. Le principe consiste à maintenir un réservoir de n éléments à partir des données du flux. Lors de l'étape d'initialisation, les n premiers éléments du flux de données sont ajoutés au réservoir. Le $i$ème élément suivant remplace de manière aléatoire l'un des éléments du réservoir avec une probabilité de $\frac{n}{i}$. La propriété de cette approche est que plus la taille du flux augmente plus la probabilité d'inclusion est réduite. Si la taille de la fenêtre est connue par avance, c'est-à-dire que le nombre d'éléments dans la fenêtre est connu, une variété de l'échantillonnage réservoir peut être utilisée nommée échantillonnage séquentiel [112]. Il s'agit de générer des sauts à effectuer entre deux inclusions. Cette méthode permet d'obtenir efficacement un échantillon aléatoire uniforme avec une complexité inférieure à la méthode standard.

On peut aussi appliquer un échantillonnage stratifié à une fenêtre fixe. On divise celle-ci en strates disjointes et on échantillonne à une taille donnée à l'intérieur de chaque strate. Le schéma le plus simple spécifie des strates de tailles approximativement égales et construit un échantillon réservoir de même taille à l'intérieur de chaque strate. Ceci permet d'éviter le cas où l'échantillonnage réservoir sous-représente les éléments de la fenêtre par exemple en sélectionnant les éléments de passé lointain par rapport au passé proche.

Une approche incrémentale de type réservoir a été présentée dans [8]. Cet article s'intéresse à la faisabilité de la constitution d'un échantillon biaisé sur des flux de données pour privilégier les éléments récents du flux par rapport aux éléments anciens. Il est montré qu'avec certaines fonctions de biais (fonctions exponentielles par exemple), l'extension de l'algorithme réservoir est possible.

Une autre méthode adaptée au flux de données est le tirage de Bernoulli [76]. Cette méthode a le mérite d'être simple à mettre en oeuvre mais a un inconvénient majeur qui est la variabilité incontrôlable de la taille de l'échantillon. Son principe consiste à effectuer une 'loterie' sur chaque individu de la fenêtre indépendamment d'un individu à l'autre. Ainsi, pour une fenêtre de taille N où les probabilités d'inclusion individuelles p_i sont connues pour tout i, on génère n réels aléatoires u_i entre 0 et 1 et on retient l'individu i si $u_i < p_i$.

StreamSamp est un algorithme de résumé généraliste basé sur l'échantillonnage de flux de données. Il a été développé par Csernel [48] pour palier aux limites de CluStream. A la différence de ce dernier, StreamSamp remplit trois critères : il est à la fois généraliste, petit et rapide dans son élaboration. Son principe est basé sur deux étapes :

1. Traitement en ligne : Un échantillonnage à un taux α fixé est effectué sur les éléments du flux. Quand une taille T de l'échantillon est atteinte, celui-ci est stocké sur disque avec ses dates de début et de fin et un ordre 0 qui lui est

associé. Puis, on procède à un second échantillonnage sur le flux. Comme il est impossible de stocker à l'infini des échantillons, une technique basée sur des fenêtres inclinées a été proposée. En pratique, quand le nombre d'échantillons d'un ordre o a atteint un certain seuil limite L, les deux échantillons de cet ordre les plus anciens sont fusionnés pour former un échantillon de taille T et d'ordre $o + 1$. Cet échantillon couvre la même période de temps que ses deux échantillons parents mis bout à bout et il est construit en sélectionnant aléatoirement $T/2$ éléments de chacun de ses échantillons parents.

2. Traitement hors ligne : Cette étape est une exploitation du résumé créé en ligne. Ainsi les échantillons d'ordres différents sont fusionnés pour reconstituer le flux de données sur une période donnée. Pour reconstituer tout le flux, il suffit de fusionner la totalité des échantillons stockés. L'ordre est pris en compte comme pondération afin de rendre l'échantillon final le plus représentatif possible. Celui-ci devient donc exploitable pour toute application des techniques classiques de fouille de données.

Grâce à l'échantillonnage, StreamSamp ne dépend pas du taux d'arrivée des flux. Ainsi, plus le taux augmente, plus les éléments sont pris dans l'échantillon. En revanche, les performances de StreamSamp se dégradent avec l'ancienneté. En effet, les anciens éléments ont un poids croissant pour une taille fixe de l'échantillon. Par conséquent, si les anciens éléments sont inclus dans un échantillon contenant des données de poids beaucoup plus faible, ils accroissent dramatiquement la variance de l'échantillon.

Échantillonnage dans une fenêtre glissante

L'échantillonnage réservoir présenté à la sous-section précédente est utile dans le cas d'insertion ou de mises à jour mais trouve ses limites à l'expiration des données dans une fenêtre glissante. En effet, dans ce type de fenêtre, les éléments ne faisant plus partie de la fenêtre courante deviennent invalides, et s'ils appartiennent à l'échantillon, il faut les remplacer. Des algorithmes permettant de tenir à jour un échantillon sur une fenêtre glissante tout en préservant sa représentativité sont donc nécessaires. Plusieurs techniques ont été développées pour traiter le cas des fenêtres glissantes (logiques et temporelles).

Une approche qualifiée de 'simple' a été proposée dans [21]. Son principe est le suivant : on maintient un échantillonnage réservoir pour les premiers éléments du flux (première fenêtre). Et quand un élément expire, on le remplace par le nouvel élément qui arrive. Cet algorithme maintient un échantillon aléatoire uniforme pour la première fenêtre et ne requiert pas beaucoup de mémoire, mais a l'inconvénient d'être hautement périodique. Pour remédier à ce défaut, une technique a été proposée : échantillonnage avec réserve. Son principe est le suivant : quand un élément arrive on l'ajoute avec une probabilité donnée à un échantillon réserve ('backing sample') et on génère ensuite un échantillon aléatoire à partir de l'échantillon réserve. Quand un élément expire, on le supprime de la réserve.

Plusieurs autres algorithmes ont été développés, applicables aux fenêtres logiques (définies dans la section 2.3) tel que l'échantillonnage en chaîne [21]. Chaque élément

i est ajouté à l'échantillon avec une probabilité de $\frac{1}{min(i,n)}$, n étant la taille de la fenêtre. Comme chaque élément est ajouté à l'échantillon, il faut choisir l'index de l'élément qui le remplacera quand il expire. On choisit l'index du « remplaçant » dans l'intervalle $[i+1, i+n]$. Quand l'élément avec cet index arrive, on l'intègre à l'échantillon et on choisit de la même façon que précédemment l'élément qui pourrait le remplacer. Ainsi, on construit une chaîne de remplaçants potentiels. Quand un élément est choisi pour être éliminé de l'échantillon, on efface sa chaîne aussi.

Le nombre d'éléments dans une fenêtre physique (définie dans la section 2.3) varie avec le temps. On ne peut donc pas connaître a priori le nombre d'éléments en avance. Plusieurs éléments dans l'échantillon peuvent expirer en même temps. Aucune des méthodes présentées précédemment ne peut être appliquée à une fenêtre physique puisqu'elles requièrent la connaissance de nombre d'éléments dans une fenêtre. Un échantillonnage par priorité, permet de contourner le problème en assignant aléatoirement pour chaque élément une priorité entre 0 et 1. Pour de plus amples informations sur les techniques présentées ici, le lecteur peut se référer à [76]. D'autres techniques ont été implémentées pour des conditions particulières d'utilisation (tel que le réservoir avec fréquences exactes [67] ou encore pour construire des échantillons de façon distribuée dans un environnement de flux de données (tel que l'échantillonnage min-wise [94]). Nous présenterons ces méthodes au chapitre 4.

3.4 Conclusion

Au cours de ce chapitre, nous avons présenté les principales approches de création de résumés à partir d'un flux de données individuel. Comme nous allons le voir dans le chapitre 6, la plupart de ces approches sont applicables à la problématique traitée au cours de cette thèse. En effet, il s'agit de résumer des flux de données provenant de multiples sources. De façon évidente, nous pouvons appliquer individuellement des techniques de résumé à chacun des flux de données. D'où la nécessité d'établir un état de l'art des approches existantes dans ce contexte.

De nombreuses structures de résumés ont été développées ces dernières années comme les sketchs, l'échantillonnage, les ondelettes et le micro-clustering. Toutes pouvant être adaptées à plusieurs types d'application avec une robustesse et une grande efficacité mémoire. De plus, ces méthodes ont été étendues aux cas des flux de données et ne nécessitent donc qu'une passe sur les éléments. Toutefois, la recherche reste très active pour améliorer quelques techniques de résumé encore très jeunes comme pour le cas du clustering. En effet, le nombre des techniques de clustering offrant des résumés généralistes est très limité. Elles sont toutes en cours de développement et n'ont pas atteint une maturité leur permettant d'être citées comme référence.

Mis à part l'échantillonnage et le clustering, une des limites des techniques de résumé citées dans ce chapitre est leur incapacité à s'adapter à des données multidimensionnelles faisant intervenir plusieurs mesures à chaque instant. Plusieurs applications génèrent de telles données. Par exemple, un capteur peut générer des mesures de température et de pression à chaque instant. Il existe peu de travaux qui se sont

intéressés à cette problématique [50]. Ces travaux peuvent être de deux types : (1) algorithmes individuels où les techniques de résumé sont effectuées séparément sur chaque mesure, (2) et les algorithmes combinés construisant le résumé en traitant toutes les données mesurées comme un seul vecteur de valeurs. De plus avec l'apparition de nouvelles applications faisant intervenir des flux de données distribués, tels que les réseaux de capteurs, des travaux [45] commencent à s'intéresser aux techniques de résumé dans un environnement distribué.

Chapitre 4

Résumé de flux de données distribués

Sommaire

4.1 Introduction

Les sources de données distribuées (aussi appelées capteurs) envoient des mesures en temps réel et à fréquence variable à un serveur central. Les applications actuelles des approches de traitement de flux de données distribués portent surtout sur la supervision de systèmes, sur le déclenchement d'alarmes en temps réel, ou plus généralement sur la production de synthèses d'aide à la décision à partir de plusieurs flux. Bauzer et al.[24] abordent les problèmes rencontrés lors du traitement de données spatio-temporelles, mesurées en temps réel, où les sources de données correspondent à des capteurs qui sont placés sur les axes routiers d'une ville et transmettent à des centrales de données des mesures sur la circulation à chaque intervalle de temps. Un autre exemple de réseaux de capteurs est donné dans [71], il concerne les technologies RFID (Radio Frequency IDentification) qui fournissent l'identification numérique des marchandises servant à les dépister. Dans un exemple cité dans cet article, on estime

les données récoltées par RFID à 300 millions de n-uplets par jour sous la forme (identifiant, date, position) si un détaillant possède 3000 magasins vendant 10000 produits par jour et par magasin (en moyenne un produit parcourt 10 emplacements avant sa vente). Ce grand volume de données ne pouvant pas être stocké entièrement, il est donc nécessaire d'effectuer un prétraitement à la volée avant leur enregistrement.

Dans le cadre de cette thèse, nous traitons un grand nombre de capteurs alimentant un serveur central. Celui-ci dispose d'un espace de stockage limité et ne peut pas accepter la totalité des données. Afin de résoudre ce problème, il est essentiel de mettre en place des techniques incrémentales d'approximation dont la qualité de résultat sera bornée en fonction des ressources disponibles (ou l'inverse : i.e. dimensionner les ressources en fonction de la qualité de l'approximation souhaitée). Plusieurs travaux de recherche ont été menés pour développer des algorithmes de résumés des données au fil de l'eau appliqués à des flux de données distribués. Quelques travaux échantillonnent les sources de flux de données (*échantillonnage d'individus* ou *échantillonnage spatial*). Dans ce cadre, on travaille sur la base d'une population finie. On construit une stratégie d'échantillonnage sur cette base afin d'utiliser au mieux les informations disponibles sur les individus pour bien estimer, sur l'échantillon qui sera finalement construit, la ou les caractéristiques que l'on souhaite mesurer. Cette technique est expliquée dans la section 4.2. D'autres travaux échantillonnent les données provenant de chaque source (*échantillonnage temporel*), ils sont présentés dans la section 4.3.

4.2 Echantillonnage spatial

4.2.1 Généralités

Nous appelons échantillonnage spatial (ou sondage [16; 108]) l'étude d'une population en examinant une partie de celle-ci. Il vise à étudier partiellement une population donnée en vue d'extrapoler à la totalité de celle-ci. L'échantillonnage spatial permet de construire un échantillon en ne prélevant qu'une partie des individus existants. Il s'oppose par cette définition au recensement qui est l'observation exhaustive et l'examen de toute la population.

Les principaux problèmes qui se posent quand on veut effectuer un échantillonnage spatial sont : (1) la sélection de l'échantillon d'individus, appelé aussi « unités d'échantillonnage » en s'interrogeant sur la méthode de tirage à employer et (2) l'agrégation des réponses qui ont été recueillies auprès des individus en déterminant les estimateurs. Il s'agit d'expressions mathématiques qui permettent de proposer, lorsque les données sont collectées, une valeur pour l'ensemble de la population en fonction des valeurs prises par les individus interpellés. Ensuite, le sondeur doit se poser la question de la précision : que vaut la procédure d'agrégation, c'est-à-dire jusqu'à quel point peut-on avoir confiance dans les résultats numériques obtenus par estimation ? En effet, le sondeur essaie de faire presque aussi bien que le recensement mais avec un coût beaucoup plus faible.

Objectifs d'un échantillonnage spatial

Soit une population U composée de N individus ou éléments appelés unités statistiques (ensemble des compteurs électriques par exemple). La taille N de la population est supposée finie et connue. On suppose donc que l'on peut dresser une liste exhaustive des éléments de la population U, appelée base de sondage, et chaque élément est représenté soit par son nom soit par un numéro (identifiant) compris entre 1 et N.

Le but du sondage est de déterminer auprès de chaque individu de la population U la valeur d'une ou de plusieurs variables dites d'intérêt. Soit C une variable d'intérêt, l'identifiant de l'individu « courant » de la population sera noté i, et la valeur prise par la variable C sur cet individu sera notée C_i. La variable d'intérêt peut être aussi sous forme d'un vecteur ou d'une série temporelle telle que la courbe journalière de consommation électrique de chaque compteur.

La fonction des C_i qui nous intéresse est un paramètre, c'est-à-dire une grandeur fixée mais inconnue. Notons la $\theta = f(C_1, C_2, ..., C_N)$. Dans le cas d'une moyenne par exemple :

$$f(C_1, C_2, ..., C_N) = \frac{1}{N} \sum_{i=1}^{N} C_i$$

Nous l'estimons grâce à une fonction g calculée à partir des données de l'échantillon seulement de taille n : $\hat{\theta} = g(C_1, C_2, ..., C_n)$. La procédure qui permet de passer des données recueillies sur l'échantillon à la vraie valeur inconnue s'appelle l'inférence statistique. On peut aussi parler d'extrapolation.

Les mesures des erreurs d'échantillonnage

Un des points fondamentaux est la nature de l'aléa introduit dans l'échantillonnage. L'échantillon d'individus interrogés a une composition aléatoire. L'estimateur $\hat{\theta}$ est donc aléatoire et est fonction de l'échantillon S. L'échantillonnage fait appel essentiellement à trois notions : le biais, la variance, et l'erreur quadratique moyenne.

- Biais : erreur systématique qui conduit à un manque de validité des données, il se calcule grâce à l'espérance $E(\hat{\theta})$ correspondant à la moyenne de $\hat{\theta}$ (en probabilité)

  $$B(\theta) = E(\hat{\theta} - \theta) = E(\hat{\theta}) - \theta$$

  $E(\hat{\theta}) = \sum_S p(S).\hat{\theta}(S)$ tel que $p(S)$ la probabilité de tirer l'échantillon S.
- Variance : grandeur liée à la dispersion des valeurs de θ. C'est la moyenne des carrés des écarts à la moyenne

  $$V(\hat{\theta}) = E(\hat{\theta} - E(\hat{\theta}))^2$$

  En terme de sondage, l'écart type ($\sigma(\hat{\theta}) = \sqrt{V(\hat{\theta})}$) et la variance mesurent la précision : plus ils sont grands, moins le plan de sondage est bon. Il faut alors soit agir sur l'expression de $\hat{\theta}$ (formule de l'estimateur), soit modifier la méthode de tirage de l'échantillon (probabilité de tirage $p(S)$).

– Erreur quadratique moyenne EQM : moyenne des carrés des écarts des estimateurs à la vraie valeur et non plus à la moyenne

$$EQM = E(\hat{\theta} - \theta)^2$$

On a aussi : EQM= VARIANCE + $(BIAIS)^2$

Bases de sondage

Pour effectuer un tirage probabiliste où chaque individu de la population a une probabilité fixée de faire partie de l'échantillon, il faut disposer d'une base de sondage répondant à plusieurs critères : chaque individu doit disposer d'un identifiant unique et précis. De même, cette base doit être exhaustive et ne comporter aucun doublon (un individu ne doit apparaître qu'une seule et unique fois dans la base). Enfin, elle doit être accessible. Mais il est possible qu'on ne puisse pas trouver de base de sondage reproduisant la population, ou qu'une telle base existe mais soit jugée de mauvaise qualité. Ou encore que la base soit trop volumineuse ou infinie (comme dans le cas d'un échantillonnage dans un flux de données individuel). Une des techniques pour détourner ce problème est de rechercher des bases de sondage non plus d'individus directement susceptibles de fournir l'information, mais de passer par un niveau intermédiaire de groupes d'individus. Par un tirage à plusieurs degrés, et en réalisant des recensements intermédiaires dans les groupes sélectionnés, on peut ainsi échantillonner rigoureusement des unités de la population qui nous intéresse.

4.2.2 Plans de sondage classique

On distingue deux grandes classes de sondage : les sondages probabilistes et les sondages empiriques. Par définition, un sondage est probabiliste si chaque individu de la population a une probabilité connue et non nulle d'appartenir à l'échantillon. Par opposition, les sondages empiriques sont ceux qui ne permettent pas de calculer la probabilité d'inclusion des individus. Ces derniers ne seront pas traités dans le cadre de cette thèse, le lecteur peut se référer au livre d'Ardilly [16] qui constitue une référence dans le domaine.

Sondage aléatoire simple

Dans un sondage aléatoire simple, tous les individus d'une population ont une chance égale d'être inclus à l'intérieur de l'échantillon. Chaque combinaison d'individus de la population a aussi une chance égale de composer l'échantillon. Un sondage aléatoire simple peut s'effectuer avec ou sans remise. Un sondage avec remise signifie qu'il est possible qu'un individu soit sélectionné deux fois ou plus pour constituer un échantillon. Habituellement, le sondage aléatoire simple est effectué sans remise, parce qu'il est plus pratique et donne des résultats plus précis.

Le sondage aléatoire simple a l'avantage d'obtenir un échantillon « représentatif » puisque la méthode donne à chaque individu de la population une chance égale d'appartenir à l'échantillon. Toutefois, pour que cela s'applique, il faut avoir une liste exhaustive de toute la population.

Sondage systématique

Parfois appelé échantillonnage par intervalles, l'échantillonnage systématique signifie qu'il existe un écart, ou un intervalle, entre chaque unité sélectionnée qui est incluse dans l'échantillon. On dispose d'un ensemble d'individus dans un ordre fixé. On calcule un nombre appelé « pas de tirage » ou « pas de sondage » : $PAS = \frac{N}{n}$

Pour constituer l'échantillon il faut choisir au hasard (aléatoirement) un entier naturel d entre 1 et PAS (cet entier sera le point de départ), l'individu dont le numéro correspond à d est le premier individu, pour sélectionner les autres, il suffit d'ajouter à d le pas de sondage : les individus choisis seront alors ceux dont les numéros correspondent à $d + PAS$, $d + 2PAS$, $d + 3PAS$ etc.

Si la population est distribuée au hasard dans la base de sondage, un échantillonnage systématique donnera des résultats similaires à ceux d'un échantillonnage aléatoire simple.

Sondage stratifié

L'idée est de partitionner la population en un nombre H fini de sous populations (mutuellement exclusifs) homogènes U_h appelées strates. L'intérêt de partitionner est de pouvoir tenir compte de la spécifité de ces sous populations dans le prélèvement de l'échantillon. On peut utiliser n'importe quelle méthode d'échantillonnage pour sélectionner l'échantillon à l'intérieur de chaque strate, la méthode d'échantillonnage pouvant varier d'une strate à une autre. Une bonne stratification donne lieu à la constitution de groupes d'individus tels que, vis-à-vis de la variable d'intérêt C, les comportements moyens soient homogènes au sein de chaque groupe et les plus différents possibles d'un groupe à l'autre.

Le problème qui se pose est de déterminer le nombre d'observations à prélever dans chaque strate, connaissant n (nombre d'éléments dans l'échantillon). Pour recourir à ce problème, on a deux modes de prélèvement. Le premier repose sur la simplicité (échantillon stratifié proportionnel avec pour chaque strate h le même taux de sondage $f_h = n_h/N_h = n/N$)et le deuxième sur l'optimalité (allocation de Newman par exemple). Un échantillon stratifié est optimal s'il comprend un nombre d'individus par strate rendant la variance globale de l'estimateur minimale. Ce nombre est proportionnel au produit du nombre d'individus par strate dans la population et de l'écart-type dans la strate. L'effectif n_h d'un sous échantillon est d'autant plus élevé que la variance de la strate h est grande, que son effectif N_h est élevé et que le coût unitaire d'échantillonnage d'une strate est faible [16]. Le coût unitaire d'une strate correspond au coût moyen de la collecte plus le traitement consécutif au tirage d'un individu de la strate. Le coût du traitement est compris de façon générale et englobe des dépenses aussi diverses que le traitement informatique en usage de ressources matérielles, etc.

La stratégie de sondage stratifié se base sur des informations disponibles sur la population étudiée. Elle permet de réduire la variance et donc d'augmenter la précision de l'estimation à condition que la stratification soit faite selon un critère effectivement lié au phénomène étudié. En effet, si tel est le cas, la variabilité du phénomène

est peu importante à l'intérieur des strates et importante entre les strates. Elle permet également, en constituant une strate à partir d'un sous-groupe de population de taille réduite, de s'assurer que ce sous-groupe est bien représenté dans l'échantillon final. Cela peut permettre également d'estimer des paramètres pour chaque strate, à condition que la taille d'échantillon à l'intérieur de chaque strate soit de taille suffisante. Cependant, cet objectif nécessite souvent d'augmenter la taille de l'échantillon total.

Sondage à plusieurs degrés

On utilise une succession de regroupements des unités statistiques pour tirer l'échantillon. On commence par constituer une partition très grossière de la population, celle-ci est partagée en sous-ensembles appelés unités primaires (up). Puis, on tire des éléments parmi les (up), à ce stade, les individus tirés au sein des (up) sont appelés unités secondaires (us) parmi lesquelles on fait un deuxième prélèvement, etc. A noter que la stratification est un cas particulier de sondage à deux degrés qui consiste à prélever un échantillon dans toutes les unités primaires. A chacun des degrés, les méthodes de sondage peuvent être utilisées (par exemple tirage proportionnel à la taille au premier degré, donc à probabilités inégales, tirage aléatoire simple au deuxième degré,etc.).

Un cas particulier est le « sondage en grappes », ce sont les sondages à plusieurs degrés (souvent deux degrés) où l'ensemble des unités au dernier degré de tirage est enquêté.

Le sondage à plusieurs degrés permet de résoudre le problème de l'absence d'une base de sondage, on peut se contenter d'un travail partiel d'établissement de cette base de sondage : seule la connaissance exhaustive des unités primaires est nécessaire. Par contre, le sondage à plusieurs degrés est, en général, moins précis que le sondage à un seul degré, pour une taille donnée de l'échantillon (en nombre d'unités statistiques au dernier degré de tirage). Ceci est dû aux « effets de grappe » : les unités statistiques regroupées dans une même unité ont souvent tendance à se ressembler, à avoir des caractéristiques communes. Le fait de concentrer l'échantillon sur un échantillon d'unités primaires peut conduire à une certaine « redondance » de l'information sur ces unités et un certain « manque de représentativité » de l'ensemble. On peut établir que la majeure partie de la variance des estimateurs dans le cas de tirages à plusieurs degrés provient souvent du premier degré de tirage.

Sondage à probabilités inégales

L'usage de sondage à probabilités inégales est particulièrement intéressant lorsque la plupart des variables sont liées par un effet de taille. Dans certains cas, on peut décider d'accorder à certaines unités une probabilité plus forte d'être sélectionnées que d'autres. Exemples :

1. Pour des enquêtes auprès des entreprises, on peut tirer les unités avec une probabilité proportionnelle, par exemple, à leur nombre de salariés, à leur chiffre d'affaires etc.

2. Pour estimer la production d'un secteur que l'on sait assurée par 2 géants du secteur et des centaines de PME, il est légitime de sélectionner d'office les 2 grandes entreprises et échantillonner de manière aléatoire quelques PME.
3. Le sondage à probabilités inégales est souvent utilisé au premier degré d'un tirage à plusieurs degrés :
   – Tirage de communes avec probabilité proportionnelle à leur population
   – puis tirage de ménages ou d'individus au deuxième degré.

Echantillonnage équilibré

Un échantillon est dit équilibré sur une ou plusieurs variables auxiliaires disponibles dans la base de sondage, lorsque pour chacune d'entre elles, l'estimateur de Horvitz-Thompson du total coïncide exactement avec le vrai total issu de la base de sondage. L'estimateur d'Horvitz-Thompson du total d'une variable se calcule en multipliant chaque valeur individuelle observée sur l'échantillon par un coefficient d'extrapolation à la population entière (ou poids de sondage) égal à l'inverse de sa probabilité d'inclusion. Un échantillon S équilibré sur la variable de contrôle X doit donc respecter la contrainte suivante :

$$\sum_{k \in S} \frac{X_k}{\pi_k} = \sum_{k=1}^{N} X_k$$

où pour tout individu k de la base de sondage (k = 1 à N), π_k désigne sa probabilité d'être sélectionné dans S et X_k la valeur qui lui est associée pour la variable X.

Les échantillons équilibrés offrent deux avantages majeurs. Par définition, ils « représentent » bien la population au regard de l'information auxiliaire choisie : ils en assurent des estimations exactes, donc non soumises à la variance d'échantillonnage. En outre, ils peuvent améliorer de manière notable la précision des estimateurs de paramètres issus du sondage, surtout si les variables d'équilibrage sont choisies avec soin. En effet, s'il existe une corrélation entre l'information auxiliaire et la variable d'intérêt, on peut assez naturellement imaginer que l'estimation du total de la variable d'intérêt sera, elle aussi, bien retranscrite par l'échantillon.

Toutefois, de nombreuses raisons contribuent à rendre l'équilibrage impossible : les problèmes d'arrondis ou l'abus de variables de contrôle, obligent souvent à se contenter d'un équilibrage « approché ». Par exemple, dans une population composée de 100 individus dont la moitié est de sexe féminin, aucun échantillon aléatoire simple sans remise de taille impaire, 15 par exemple, ne peut être exactement équilibré sur le sexe et restituer le nombre exact de femmes sinon il faudrait sélectionner un effectif non entier de femmes. La conséquence de cette approximation devient négligeable pour de grands échantillons.

Avantages et inconvénients

Chaque méthode comporte des avantages et des inconvénients. Le tableau 4.1 qui suit en présente un résumé [5]. Les méthodes présentées dans les sections précédentes

sont loin d'être antinomiques : on a même vu qu'elles s'utilisent souvent en complément les unes des autres. Le choix de la méthode de sondage s'appuie d'abord sur des critères théoriques (objectifs de l'étude). Par ailleurs, toute connaissance préalable du phénomène étudié, devrait si possible être utilisée : une variable connue comme liée à l'objet de l'étude permettra ainsi de choisir un type de sondage plus représentatif qu'un sondage aléatoire simple. En pratique cependant, ce sont les critères de faisabilité qui sont souvent les plus déterminants pour le choix de la méthode, en particulier le type de base de sondage accessible, complète, exacte et à jour.

4.2.3 Estimateur Horvitz-Thompson

En 1952, Horvitz et Thompson ont présenté un estimateur linéaire sans biais d'un total T valable pour tout plan de sondage :

$$T_\pi = \sum_{k \in S} \frac{C_k}{\pi_k}$$

Cet estimateur est appelé estimateur d'Horvitz-Thompson ou π-estimateur. Cet estimateur permet d'estimer la taille de la population, qu'elle soit connue ou non :

$$N_\pi = \sum_{k \in S} \frac{1}{\pi_k}$$

La variance de l'estimateur d'Horvitz Thompson est donné par :

$$Var(T_\pi) = \sum_{i \in U} \sum_{j \in U} \left(\frac{C_i}{\pi_i} \frac{C_j}{\pi_j} \frac{\pi_{ij} - \pi_i \pi_j}{\pi_{ij}} \right)$$

Avec π_{ij} la probabilité d'inclusion double qu'un couple (i, j) appartienne conjointement à l'échantillon. Cette probabilité dépend de la stratégie utilisée lors de l'échantillonnage. L'estimateur de la variance présenté ci-dessus est souvent délicat à calculer de manière exacte ; des approximations par linéarisation, ou par bootstrap sont proposées dans la littérature [55; 16].

4.2.4 Méthodes de redressement ou de pondération

Lorsqu'on dispose d'une information auxiliaire, il faut chercher à l'utiliser dans le but d'obtenir des estimations plus précises que les estimateurs simples de la moyenne ou du total. Cette information peut être utilisée au niveau du tirage de l'échantillon ou au niveau de l'expression de l'estimateur. Si plusieurs variables auxiliaires sont utilisées, on peut recourir à une technique mixte dans laquelle certaines variables servent à améliorer le tirage, et les autres à améliorer l'estimateur.

Le redressement est un procédé permettant, à partir d'une information auxiliaire, de modifier un estimateur initial de façon à ce que sa nouvelle formulation estime avec une variance nulle une expression donnée, définie seulement à partir de l'information auxiliaire et dont la valeur numérique est connue.

Echantillonnage	Avantages	Inconvénients
Aléatoire simple	Assure le caractère représentatif de l'échantillon Assez simple à employer	Long et impraticable si la population est très nombreuse Nécessite une base de sondage complète
Stratifié	Assure à chaque sous-groupe de la population une représentativité dans l'échantillon Très précis	S'appuie sur une connaissance préalable de la population Nécessite une base de sondage Subdivision peut être difficile selon les strates voulues
Périodique	Facile d'application (sélection d'un seul nombre aléatoire) Peu coûteux	Peut ne pas être représentatif s'il y a un phénomène de périodicité
Par grappes	Rapide Ne nécessite pas une base de sondage complète	Peu représentatif si les individus dans les grappes possèdent des caractéristiques semblables
A plusieurs degrés	Rapide Ne nécessite pas une base de sondage complète Réduit l'inconvénient lié à l'échantillonnage par grappes	Peu représentatif si les unités à chaque degré possèdent des caractéristiques semblables (inconvénient réduit par rapport à l'échantillonnage par grappes)
Avec probabilités inégales	Tient compte du poids de chaque individu dans la population	Application complexe
Equilibré	Assure des estimations exactes Peut améliorer la précision des estimateurs Très précis	S'appuie sur une connaissance préalable de la population Problèmes d'arrondis Programmation complexe

TABLE 4.1 – Tableau comparatif des méthodes de sondage

Pratiquement, on part d'un estimateur classique ne tenant pas compte de l'information auxiliaire et on modifie les poids des individus de l'échantillon pour obtenir la propriété voulue. Le terme redressement s'applique donc aux poids de sondage des individus de l'échantillon. On distingue trois grandes méthodes de redressement :

- Estimateur post stratifié
- Estimateur par le ratio
- Estimateur par la régression

Post stratification simple

Le principe de cette méthode est le suivant : on étudie un caractère C sur une population. On connaît un autre caractère X sur cette même population et surtout sa distribution. L'échantillon n'est pas stratifié a priori sur X mais pour chacune des unités échantillonnées on relève le couple (C_i, X_i). On définit, a posteriori, des strates selon les valeurs de X. On repondère les données par les poids véritables des strates définies sur X.

La stratification a posteriori se justifie lorsque la variable étudiée est corrélée avec le critère de stratification et que les effectifs N_h (taille de la strate h) ou les poids des post-strates sont connus.

Estimateur par le ratio (ou par le quotient)

L'estimation par le quotient est une méthode de redressement d'échantillon sur variables quantitatives. Pour atteindre notre but qui est d'estimer une moyenne, un total, un pourcentage, etc. on utilise le principe suivant :
L'échantillon fournit une estimation ponctuelle de la moyenne $\bar{C}$ par exemple. D'autre part, on connaît la moyenne échantillonnée $\bar{x}$ d'une variable auxiliaire X. Or, on connaît également la vraie moyenne $\bar{X}$ de cette variable auxiliaire X.

L'estimateur de la moyenne $\bar{C}$ par la méthode de l'estimation par le quotient est égal à : $\bar{C}_Q = \bar{C} \times \frac{\bar{X}}{\bar{x}}$
C_Q est en général un estimateur biaisé. Le biais est d'ordre de grandeur $1/n$. Le biais est donc faible si l'échantillon est grand. Malgré tout, l'estimateur $\bar{C}_Q$ peut-être plus précis que l'estimateur $\bar{C}$.

L'estimateur par la régression

Cette méthode suppose une relation de type affine entre C, la variable d'intérêt, et X, la variable auxiliaire : $C = a + bX$. L'idée va être d'estimer le paramètre b, puis d'utiliser la grandeur $\bar{X}$ (valeur moyenne de X sur la population, connue) pour redresser et fournir l'estimateur par la régression de la moyenne : $\bar{C_{reg}} = \bar{C} + \hat{b}(\bar{X} - \bar{x})$ où $\hat{b}$ est l'estimation de b par la méthode des moindres carrés ordinaires appliquée à l'échantillon.

Cette méthode suppose des calculs complexes et est peu utilisée en pratique. On utilise parfois une variante, l'estimation par la différence, où la valeur de b est choisie

a priori égale à 1 : $\bar{C}_{diff} = \bar{C} + (\bar{X} - \bar{x})$ (on ajoute à $\bar{C}$ la différence constatée entre $\bar{X}$ et $\bar{x}$).

4.2.5 Echantillonnage spatial dans le temps : Panels

Lorsqu'on cherche à mesurer l'évolution d'un paramètre au cours du temps, on tire aléatoirement un échantillon à une date d'origine et on suit les individus de l'échantillon aussi longtemps que nécessaire pour les besoins de l'étude. Un tel échantillon où les individus sont interrogés au moins deux fois, s'appelle un panel [84]. On peut utiliser des panels si le paramètre que l'on cherche à estimer fait explicitement intervenir les évolutions individuelles de la variable d'intérêt. Par exemple, si on veut estimer la proportion de personnes qui ont dépassé un seuil donné de consommation électrique et qui ont réduit leur consommation l'année suivante.

Mise en place d'un panel

En plus des décisions qui doivent être prises pour un sondage, il faut ajouter une complexité additionnelle dûe à la dimension temps afin de mettre en place un panel. Voici les principales décisions à prendre :

- Choix de l'échantillon : la première étape pour la mise en œuvre d'un panel est le choix de l'échantillon à interroger. Cette première étape est le fruit de deux processus : un processus de sélection qui consiste à constituer un échantillon représentatif de la population de base, et un processus de renouvellement permettant de s'assurer de la pérennité de la représentativité de l'échantillon.
- Durée du panel : plus le panel est de longue durée, plus les données sont riches en information. En revanche, il est difficile de garder un échantillon représentatif sur une longue durée. Il peut parfois être bénéfique de faire varier la durée du panel selon divers types d'individus.
- Nombre de vagues de collecte de données, il est déterminé par la durée du panel et par la variable à étudier.
- Rythme de renouvellement de l'échantillon : si l'on veut effectuer une analyse longitudinale (c'est à dire une analyse à différents moments dans le temps) sur le panel, il est suffisant de suivre simplement l'échantillon sélectionné au moment de la vague initiale (cas d'une cohorte). Mais, si on veut effectuer des analyses ponctuelles, à chaque vague, il est nécessaire de renouveler le panel pour assurer sa représentativité au cours du temps et tenir compte des changements dans la population. Dans le cas particulier de mesures de consommation électrique, on estime que renouveler un cinquième de l'échantillon par an est un bon compromis, qui permet de comprendre l'impact d'un changement quelconque sur le profil de consommation.

Problèmes spécifiques aux panels

Les enquêtes par panel sont confrontées, comme pour les sondages, à une vaste gamme de sources d'erreur non dues à l'échantillonnage. Tout d'abord, on peut

être confronté à la lassitude des individus faisant partie du panel. Pour limiter ce phénomène, un remède consiste à mettre en place un système de rotation de sous-échantillons panélisés. par exemple, on tire 4 sous-panels initiaux, et chaque panel a une durée limitée à quatre ans (ou quatre vagues). On tire chaque année un échantillon panel entrant, tandis qu'un échantillon panel sort. Chaque panel entrant est tiré selon le même plan de sondage à partir de la base de sondage mise à jour.

Un autre problème peut provenir de la disparition d'individus. C'est un phénomène naturel qui diminue la précision puisque la taille utile de l'échantillon diminue. Ces disparitions sont dues à plusieurs raisons : mort « naturelle », changement de catégorie (de tarif de consommation par exemple), déménagement, etc. Dans ce cas, le traitement du problème dépend principalement de la cause de la disparition de l'individu. Lorsque l'individu « disparu » du panel reste dans le champ de l'enquête, on est face à un phénomène du type « non réponse », et il convient de repondérer les individus restés dans le panel ou d'utiliser une technique d'imputation (ou encore une technique mixte). La méthode de pondération consiste à laisser de côté tous les enregistrements ayant une ou plusieurs vagues manquantes et à essayer de compenser leur absence par des rajustements de pondération appliqués aux enregistrements restants. On estime les valeurs manquantes grâce à des méthodes comme par exemple la moyenne (la valeur moyenne de l'ensemble de réponses obtenues est utilisée pour remplacer chacune des valeurs manquantes), ou le Hot-Deck (chaque valeur manquante est remplacée par une valeur prise aléatoirement et avec remise parmi les l'ensemble des répondants), etc.

4.2.6 Echantillonnage spatial de flux de données distribués

Nous présentons ici quelques travaux existants dans le domaine des flux de données qui sont basés sur l'échantillonnage spatial afin de répondre à la volumétrie des données et aux contraintes de ressources limitées. Ces travaux sont le plus souvent mis en place pour des applications faisant intervenir des capteurs à des fins de surveillance ou de contrôle telles que la surveillance de trafic ou encore la surveillance de phénomènes environnementaux ou urbains. Ces capteurs constituent le système d'acquisition, ils sont organisés ou non en réseau selon l'application, et envoient des mesures en temps réel et à fréquence variable vers un système centralisé. La volumétrie et la dimension temps réel provoquent de nouveaux besoins en termes de traitement des données distribuées, de leur exploitation et de leur stockage.

Willet et al. [114] proposent une approche, appelée « backcasting », qui peut réduire de manière significative les coûts de communications et de consommation d'énergie. Cette approche opère en activant seulement un petit sous-ensemble de capteurs qui vont communiquer leurs informations à un serveur central. Ceci fournit une estimation initiale de l'environnement et guide le serveur central à allouer des ressources supplémentaires aux capteurs (faire appel à des capteurs supplémentaires) afin d'atteindre une précision donnée. L'estimation initiale sert à détecter des éventuelles corrélations dans l'environnement afin de réduire le nombre de capteurs auxquels le serveur central fait appel. Toutefois la sélection des capteurs se fait de façon empirique et ne peut donc pas utiliser les estimations probabilistes vues précé-

demment. Une approche similaire a été développée par Hartl et al. dans [77] afin de calculer la moyenne des données dans un réseau de capteurs. En effet, on active un sous-ensemble de capteurs et on utilise leurs données afin d'inférer sur les données de capteurs 'inactifs' en utilisant l'inférence bayésienne. L'algorithme développé est appelé *infer* et opère en deux phases. Durant la première phase, un algorithme distribué décide quels capteurs doivent être actifs pour la prochaine période de temps en se basant sur leurs niveaux d'énergie (afin d'augmenter la durée de vie du réseau de capteurs) et sur leurs données du passé. La deuxième phase s'effectue au niveau du serveur central et utilise l'inférence bayésienne afin de prédire la moyenne des données de l'ensemble des capteurs. Même si les auteurs ne se sont intéressés qu'à la moyenne AVG afin de valider leur approche, l'inférence bayésienne a le mérite de pouvoir s'appliquer à un large éventail de requêtes d'agrégation.

Nath et al. [94] utilisent l'échantillonnage Min-Wise [27] afin de sélectionner uniformément et aléatoirement un certain nombre k de capteurs. Le principe de cette technique est le suivant : étant donnée une fonction de hachage h appliquée aux identifiants des capteurs, le serveur central sélectionne les k capteurs dont la fonction de hachage donne la valeur minimale. Cette technique a le mérite d'engendrer un échantillon aléatoire et uniforme et ressemble fortement à un échantillonnage de Bernoulli. Une autre technique permettant d'échantillonner spatialement des sources de flux de données distribués a été proposée par Bash et al. [23]. Les auteurs utilisent un échantillonnage aléatoire uniforme des nœuds dans un réseau de capteur afin d'agréger les données provenant des capteurs. Leur approche se base sur des données auxiliaires telles que des données géographiques afin de déterminer le nombre de nœuds à sélectionner.

4.3 Echantillonnage temporel

Nous avons vu dans la section précédente que nous pouvons réduire la volumétrie de données dans un environnement de flux de données distribués en ne conservant que certaines sources de données en utilisant des techniques de sondage, c'est le cas de l'échantillonnage spatial. Une seconde approche pouvant naturellement être utilisée est de ne conserver que certaines mesures (à certains instants) au sein de chaque source de données. Nous appelons cette approche : *échantillonnage temporel*. Ceci permet de réduire le coût de transmission des données en permettant aux sources de données de construire localement des résumés. Ainsi, le flux n'est plus composé de données brutes mais de résultats partiels d'un processus de résumé qui vont être progressivement fusionnés (selon l'architecture de l'environnement) jusqu'à l'arrivée au serveur central. La plupart des travaux présentés dans cette sous-section ont été présentés dans le tutoriel de Cormode et Garofalakis [41] autour de la fouille de flux de données distribués.

4.3.1 Hypothèses et bases de conception

Avant d'exposer les techniques existantes, nous commençons par présenter les hypothèses et les bases de conception des méthodes d'échantillonnage temporel :

- **Résultats requis :** Les méthodes d'échantillonnage temporel dépendent de l'attente de l'utilisateur. celui-ci peut demander un résultat déterministe, ce qui signifie que la réponse à une requête en utilisant un résumé doit s'approcher du résultat exact avec un certain seuil ϵ (avec une probabilité de 100%). Ou encore, l'utilisateur peut demander un résultat probabiliste, c'est-à-dire que la réponse approchée à une requête s'éloigne de la réponse exacte d'un certain ϵ avec une certaine probabilité p.
- **Mode de communication :** On distingue deux modes de communication : décentralisé (*push*) et centralisé(*pull*). Dans le mode *push*, les sources de données sont programmées afin d'envoyer régulièrement leurs données résumées au serveur central. Alors que dans le mode *pull*, les données sont collectées à partir des sources selon la requête posée. On peut aussi imaginer un troisième mode de communication hybride entre les deux précédents, où les nœuds envoient régulièrement leurs résumés au serveur central, qui pourra demander un complément d'information le cas échéant.
- **Architecture :** On distingue deux types d'architecture : plate (2 niveaux) et hiérarchique (multi-niveaux). Dans l'architecture plate, la décision des données à conserver peut être prise par les sources (premier niveau) ou par le serveur central (deuxième niveau). On retrouve un modèle qui généralise ce dernier, et qui inclut des groupes de sources de données. Ainsi le résumé peut être construit par un cluster (premier niveau) ou par le serveur central (deuxième niveau). Dans le modèle hiérarchique, les données résumées sont fusionnées par les nœuds intermédiaires par lesquels elles transitent pour arriver au serveur central.
- **Corrélation des données :** La plupart des techniques que nous allons présenter dans cette section n'exploitent pas les corrélations entre les données des capteurs pour réduire les coûts de transmission. Cependant, il existe des travaux qui ont pris en compte ces corrélations et qui concernent un ou plusieurs attributs (par exemple, des corrélations entre l'humidité et la température) tel qu'il a été décrit dans [102].

Nous présentons maintenant un aperçu des travaux récents autour de l'échantillonnage temporel des flux de données provenant de capteurs distribués. Nous organisons ces travaux en deux groupes selon les requêtes à traiter (requêtes d'agrégation ou sur données individuelles).

4.3.2 Echantillonnage temporel pour l'agrégation

Cette sous-section se concentre sur les techniques de traitement des requêtes d'agrégation dans les environnements distribués telles que AVG, SUM, MEDIAN, MIN, MAX, COUNT, ou autre fonction définie par l'utilisateur. Un opérateur d'agrégation peut ou non se calculer de façon distribuée, si c'est le cas, des résultats partiels

peuvent être calculés tout au long du chemin parcouru entre les sources des données et le serveur central. Par exemple, les opérateurs (MIN, MAX, COUNT et AVG) sont des opérateurs distribués, alors que MEDIAN ne l'est pas (nous ne considérons pas ce type d'opérateurs dans la suite de cette section).

Yu et al. [115] présentent une technique d'échantillonnage basée sur une architecture hiérarchique sous forme d'arbre. Ils proposent un protocole Adaptatif pour une Agrégation Approchée (A^3 pour Adaptative Approximate Aggregation) avec une contrainte d'erreur fixée. Dans l'architecture de l'environnement, les sources des données représentent les feuilles qui sont connectées à des nœuds intermédiaires (pères) formant des clusters. Chaque source envoie les données à son nœud père qui agrège les données et les envoie au niveau supérieur (serveur central). Le père sauvegarde localement les anciennes valeurs de chaque source et calcule une valeur par défaut ainsi que l'erreur commise. Une source n'envoie une donnée que si la valeur par défaut dépasse l'erreur fixée.

Cormode et al. [40] utilisent des résumés de quantiles afin de suivre en continu la distribution des données provenant de multiples sources distantes. Le traitement de quantiles est différent de l'approche précédente car a besoin d'utiliser la distribution de l'ensemble des données distribuées. Une erreur fixée est utilisée afin de répondre aux requêtes, elle est divisée en deux parties ($\epsilon = \phi + \theta$). Chaque source surveille son flux et si la distribution de celui-ci s'écarte du seuil défini par θ, la source envoie le quantile approché et mis à jour avec une mise à jour de l'erreur qui ne doit pas excéder ϕ. Afin de minimiser les coûts de communication, Cormode et al. utilisent aussi un modèle de prédiction dans leur approche.

Considine et al. [39] et Nath et al. [94] ont indépendamment développé des techniques de résumé temporel en présence de perte de données ou de panne de capteurs dans une architecture hiérarchique (sous forme d'arbre). Ainsi, un capteur qui devient inopérant entraîne avec lui l'inactivité d'un sous-ensemble de l'arbre. Un routage multi-chemin a été proposé afin de pallier à ce problème. Or, pour les agrégats sensibles à la duplication (tels que SUM et COUNT), cette solution peut donner des résultats incorrects. [39; 94] ont proposé des approches qui s'appliquent à ce type d'agrégats. Par exemple, Nath et al. [94] utilisent des résumés qui sont insensibles à la duplication et à l'ordre d'arrivée des données (ODI Order and Duplicate-Insensitive). La technique proposée se base sur trois fonctions dont les implémentations différent selon le type de la requête : génération $g()$ qui génère un résumé pour chaque source de données, fusion $f()$ qui fusionne deux résumés et qui l'envoie au serveur central disposant de la troisième fonction d'évaluation $e()$ pour estimer l'erreur probabiliste de la réponse approchée à la requête. Parmi les résumés ODI, les auteurs citent : le sketch de Flajolet Martin, les filtres de bloom, l'échantillonnage aléatoire uniforme et l'échantillonnage min-wise.

4.3.3 Echantillonnage temporel pour données individuelles

Nous présentons ici des techniques de résumés de flux de données distribués afin de répondre à des requêtes sur les données brutes et non sur des données agrégées. Ces

techniques prennent en compte les corrélations temporelles et/ou spatio-temporelles entre les données des flux.

Dans [87], Lazardis et al. présentent une technique de compression de séries temporelles provenant de capteurs (appelés 'producteurs' dans l'article). Un serveur central (appelé 'archiveur') se charge de collecter les données résumées avec une qualité garantie pour chaque valeur individuelle et de prédire les valeurs futures des séries temporelles en se basant sur les résumés. Les auteurs utilisent une version en ligne de l'algorithme ACP (Analyse en Composantes Principales) comme méthode de compression qui donne un résumé en escalier sous forme de segments (v_i, t_i), i.e une valeur constante v_i représentant une période donnée t_i. Comme le temps de compression peut être long, les auteurs ont ajouté un modèle de prédiction permettant de répondre aux requêtes qui nécessitent d'utiliser des valeurs non encore compressées.

Une approche concernant l'échantillonnage temporel est présentée par Jain et al. dans [79]. Dans cet article, un filtre de Kalman est utilisé au niveau de chaque capteur pour effectuer des prédictions. Une granularité temporelle (un pas d'échantillonnage) est ajustée selon l'erreur de prédiction. Si ce pas dépasse un seuil alloué par le système central, un programme d'optimisation est lancé au niveau central afin d'affecter de nouveaux seuils aux capteurs. Une approche prédictive similaire a été proposée par Liu et al. [88]. Leur approche permet d'éviter aux capteurs d'envoyer leurs données si le serveur central peut les prédire en utilisant un modèle ARIMA dont les paramètres sont partagés entre le serveur central et chaque capteur.

Une autre technique pour la compression de données historiques dans un réseau de capteurs a été présentée par Deligiannakis et al. [49]. Ils supposent que de multiples flux de données sont produits par chaque capteur (par exemple, un flux de température et un autre pour la pression) et sont envoyés à un serveur central. L'objectif des auteurs est d'exploiter les corrélations entre les flux de données de chaque capteur et aussi entre les données d'un seul flux (corrélation temporelle telle que la périodicité). Tout d'abord, chaque capteur construit un signal de base d'une taille fixée. Il s'agit d'une référence ou d'un dictionnaire permettant de décrire les caractéristiques du flux de données et de l'approcher grâce à une méthode de régression. L'algorithme proposé est appelé SBR (pour Self-Based Regression) : il permet de construire une meilleure compression d'une série temporelle prenant en compte les contraintes d'espace de stockage et les tailles des signaux de base. Les auteurs ont évalué leur approche en utilisant plusieurs méthodes de compression telles que les ondelettes, les histogrammes et la transformation Cosinus discrète (DCT) sur des jeux de données réels.

Les deux prochaines méthodes que nous allons présenter visent à résumer les données provenant de multiples capteurs en prenant en compte à la fois les corrélations temporelles et les corrélations spatiales (dans le sens géographique du terme). Deshpande et al. [52] proposent un modèle statistique basé sur des gaussiennes multivariées afin de réduire les coûts de communication dans un réseau de capteurs. La fonction de densité du modèle est estimée lors d'une première phase d'apprentissage en utilisant les données historiques. Le modèle appelé BBQ (pour Barbie-Query) tourne au niveau du serveur central auquel les utilisateurs envoient des requêtes SQL incluant des intervalles de confiance désirés. Les requêtes considérées dans ce travail

sont des requêtes ad-hoc envoyées régulièrement et non pas des requêtes continues. Le serveur central utilise le modèle gaussien afin de vérifier sa possibilité de répondre à une requête avec l'intervalle de confiance fixé. Si ce n'est pas le cas, la méthode se base sur les corrélations spatiales et temporelles afin de déterminer les capteurs et les attributs susceptibles de répondre correctement à la requête tout en minimisant les coûts de communication.

Chu et al. [38] traitent les requêtes de sélection sous forme 'SELECT *' dans un réseau de capteurs. Le prototype proposé appelé Ken utilise un modèle probabiliste dynamique partagé entre le serveur central et chaque nœud du réseau. Ken ressemble beaucoup au prototype BBQ proposé par Deshpande et al. [52], dans le sens où Ken estime aussi un modèle à partir des données historiques et prend en compte les corrélations spatiales et temporelles intra et inter-capteurs. Leur principale différence réside dans la façon d'acquérir les données au niveau du serveur central pour répondre aux requêtes. L'acquisition des données dans BBQ est de type 'pull', c'est-à-dire que le serveur central ne reçoit les données du réseau que lorsque cela est nécessaire afin de satisfaire les requêtes. Tandis que l'acquisition des données dans Ken est de type 'push' où les nœuds surveillent leurs données continuellement et les envoie uniquement quand les valeurs dévient du modèle probabiliste partagé avec le serveur central. Ken utilise les corrélations spatiales en partitionnant (grâce à des heuristiques) les sources de données en clusters et en construisant un modèle probabiliste commun à tous les nœuds d'un même cluster.

4.4 Conclusion

Nous avons vu tout au long de ce chapitre qu'il existe deux principales approches visant à réduire la charge du serveur central qui collecte des données provenant de multiples sources distribuées et à minimiser les coûts de transmission. Il s'agit de :

- L'approche spatiale qui consiste à conserver un certain nombre de sources de données constituant un panel. La première étape pour la mise en oeuvre d'un panel est le choix de l'échantillon à suivre. Cette première étape est le fruit de deux processus : un processus de sélection qui consiste à constituer un échantillon représentatif de la population de base, et un processus de renouvellement permettant de s'assurer de la pérennité de la « représentativité » de l'échantillon. Pour assurer cette « représentativité », il faut tout d'abord définir les objectifs du sondage. Il faut s'interroger ensuite sur la taille de l'échantillon et procéder à la sélection en prenant en compte quelques critères (variables auxiliaires corrélées à la variable d'intérêt qui est ici le flux de données). Dans ce cadre, nous avons tout d'abord défini les notions liées au sondage et à la constitution de panel et ensuite nous avons cité les travaux appliquant cette approche dans un contexte de flux de données distribués.
- L'approche temporelle qui consiste à conserver un nombre réduit de mesures de chaque flux de données individuel et ceci au niveau de toutes les sources de données. Ainsi, toutes les sources sont complètement ou partiellement observées contrairement à l'approche spatiale. Ceci permet de répondre à un plus vaste

choix de requêtes. Les techniques de résumé utilisant l'approche temporelle dépendent de plusieurs paramètres tels que l'architecture de l'application, le mode de communication et le type de requête à traiter. Elles exploitent aussi parfois les corrélations qui pourraient exister entre les données d'une même source ou entre les données de sources différentes afin de réduire les coûts de communication et de stockage.

La principale question qui se pose à nous à l'issue de ce chapitre est la suivante : quelle stratégie (temporelle ou spatiale) utiliser afin de réduire la volumétrie des flux de données distribués ? Il est évident que la réponse à la question dépend de l'application et donc des traitements qu'on veut effectuer aux données. Nous tâcherons de répondre à cette question dans la suite de cette thèse dans le cadre de traitement de requêtes d'agrégation (SUM, AVG, MEDIAN, etc.) sur tout ou partie des flux de données distribués. Nous proposerons aussi plus loin une stratégie qui consiste à combiner ces deux approches, par souci d'avoir des estimateurs des flux agrégés de qualité meilleure que si on avait effectué une approche purement temporelle ou purement spatiale.

Chapitre 5

Application expérimentale

Sommaire

5.1 Définition de la problématique

La connaissance du profil de consommation des clients est très importante pour une société comme EDF (Electricité De France). En effet, cette connaissance permet de proposer des offres tarifaires adaptées à chaque client, d'anticiper les besoins électriques, de réagir sur la demande en cas de situation critique, etc. Pour cela, outre les données marketing dont il dispose, le fournisseur d'électricité relève des index de consommation de ses clients afin d'analyser les courbes de charge. La courbe de charge correspond à l'évolution de la consommation électrique en puissance à un intervalle de temps allant de la seconde à 30 jours pendant une période de temps défini.

Aujourd'hui, EDF dispose des index de consommation électrique relevés mensuellement chez ses clients. De plus, une campagne de mesures permanentes est menée auprès d'un panel de clients. EDF dispose ainsi de données de consommation pour un échantillon de clients bien choisis et représentatif de la population totale, constituée de plus de 30 millions de clients. L'élaboration d'un panel est problématique car il faut 'recruter' des individus pour faire partie du panel, équiper les foyers pour la relève de la consommation et ensuite maintenir le panel à jour (renouvellement de l'échantillon) pour assurer dans le temps la collecte de données représentatives de la population cible.

Il est envisagé de déployer en France dans les prochaines années des compteurs communicants chez un grand nombre, voire la totalité, de clients d'EDF, à l'image de ce qui a été déjà réalisé en Italie par la société ENEL. Ces nouveaux compteurs,

reliés aux Systèmes d'Information de la société, permettront de relever à distance les consommations d'électricité afin d'effectuer des opérations telles que la facturation, l'agrégation des courbes de charge, le contrôle de la consommation globale, la supervision du réseau, etc. La relève se fera quotidiennement et non plus mensuellement et générera un flux de données extrêmement volumineux. Le nombre de compteurs électriques s'élève à plus de 30 millions, on ne peut évidemment pas tous les analyser. Une estimation des volumes en jeu nous donne comme ordre de grandeur 1 Mo/courbe/an, soit 30 To/an pour les 30 millions de compteurs, pour des données collectées toutes les 5 minutes.

Si ce nouveau système permettra toujours l'utilisation de panels standards de courbes de charge telle qu'aujourd'hui, il offre également d'autres opportunités, dont la mise en place de panels « souples » et dynamiques ou l'application de techniques d'échantillonnage temporel qui permettront, à terme, de piloter dynamiquement la fréquence de télé-relève des clients en fonction des contraintes de l'infrastructure et des besoins en terme d'analyse. Il paraît donc intéressant de réfléchir à des méthodes permettant de mettre en oeuvre des techniques de résumé dans un environnement distribué.

5.2 Données mises à disposition

Les données exploitées sont des données de consommation relevées par les compteurs électriques. Le schéma des enregistrements est constitué de trois champs. Le premier est une estampille temporelle donnant la date du relevé, le second est une mesure de consommation électrique en puissance et le dernier est l'énergie consommée à chaque instant. A partir des données réelles, nous avons modélisé plusieurs flux de données correspondant à des mesures de consommation électrique générées régulièrement et avec le même taux d'arrivée et au même instant (timestamp) par chaque capteur (compteur électrique). Le tableau 5.1 ci-dessous montre un exemple de flux de données généré par un compteur électrique avec un intervalle de temps de 30 minutes entre deux mesures consécutives.

Timestamp	Puissance (kW)	Energie (kWh)
...	...	...
16/12/2006-17 :00	19.67	9.84
16/12/2006-17 :30	16.33	18
16/12/2006-18 :00	15.33	25.67
16/12/2006-18 :30	16.33	33.83
...	...	...

TABLE 5.1 – Exemple de flux de consommation électrique

Pour mener à bien les travaux de résumé de flux de données distribués, il nous a été fourni un ensemble de jeux de données. Il s'agit de courbes relatives à de petites entreprises réparties sur toute la France. Les courbes ont été pré-traitées et

ne présentent pas de valeurs manquantes. Chaque client possède un identifiant, un numéro de 1 à N (N étant la taille du jeu de données en nombre de compteurs).

5.2.1 Premier jeu de données

Le premier jeu de données correspond à $N = 140$ compteurs électriques chacun produisant une mesure toutes les 30 minutes et ceci pendant 365 jours (un an). Nous avons reçu ce jeu de données sous forme d'un seul fichier texte pour chaque compteur correspondant à sa courbe de charge pendant les 365 jours. Nous avons traité les fichiers afin de simuler des flux de données correspondant à des relevés de consommation électrique par jour et par semaine. Ce qui signifie que nous disposons de 48 mesures par jour et 336 mesures par semaine et ceci pour chaque capteur. La figure 5.1 montre un exemple de deux courbes de charge, la première prise pendant une journée constituée de $p = 48$ points et la seconde pendant une semaine constituée de $p = 336$ points. On remarque une périodicité dans l'évolution de la consommation électrique dans la seconde courbe correspondant à la courbe de charge d'une semaine. Cette périodicité sera exploitée dans nos approches que nous verrons dans la suite de cette thèse.

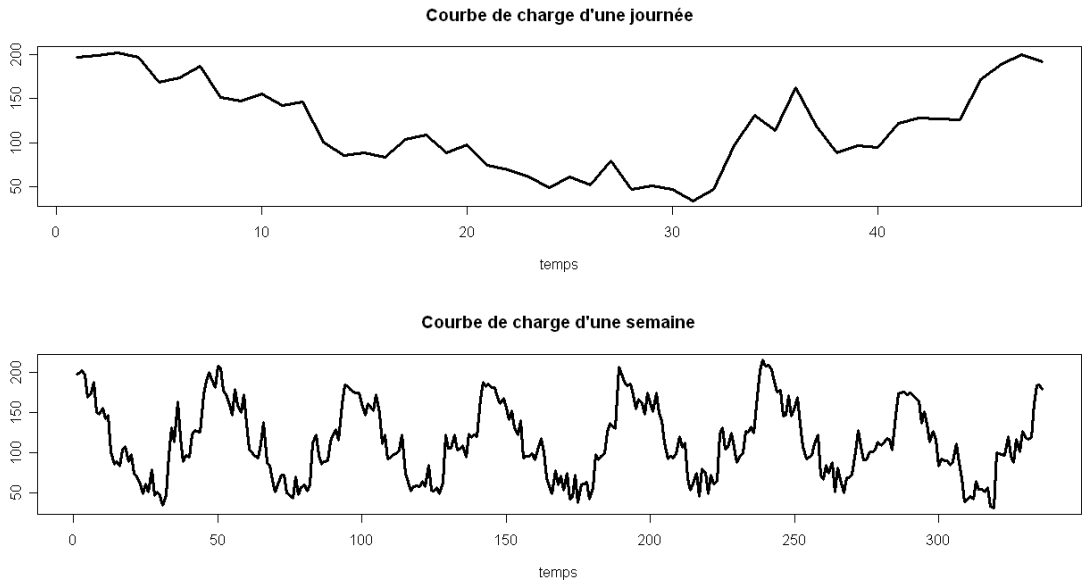

FIGURE 5.1 – Un exemple de compteur du premier jeu de données

5.2.2 Second jeu de données

Le second jeu de données correspond à $N = 1000$ courbes de charge correspondant à une période d'un an et mesurées à des intervalles de temps de 10 minutes. Comme pour le premier jeu de données, nous avons traité les fichiers reçus afin de modéliser

l'arrivée de 1000 flux de données découpés en période d'une journée (144 mesures par capteur) et d'une semaine (1008 mesures par capteur). La figure 5.2 montre un exemple de deux courbes de charge, la première prise pendant une journée ($p = 144$ points) et la seconde pendant une semaine ($p = 1008$ points).

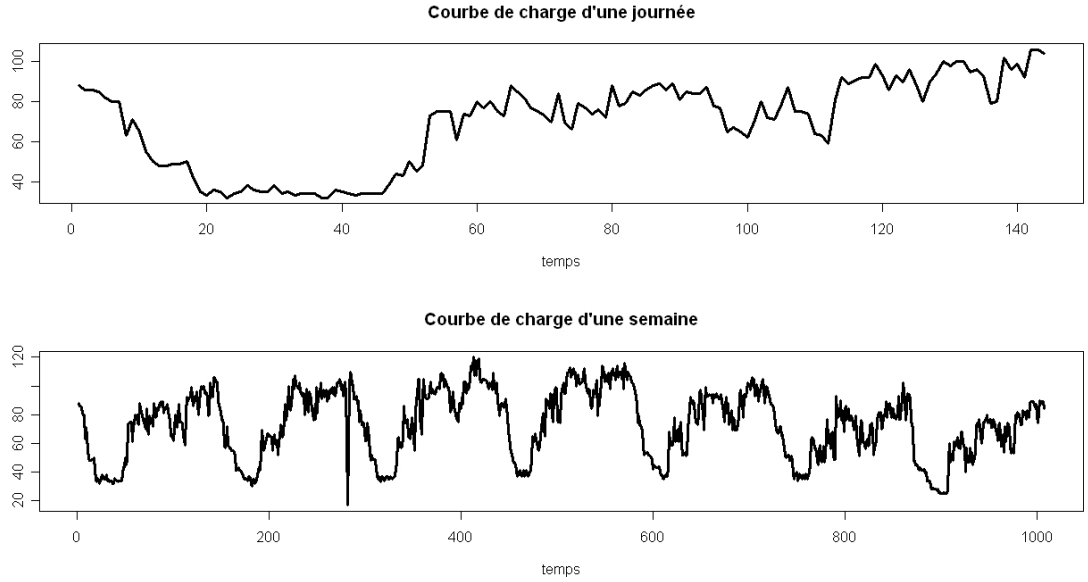

FIGURE 5.2 – Un exemple de compteur du second jeu de données

5.3 Conclusion

Dans ce chapitre ont été présentés les jeux de données que nous allons utiliser dans la suite de cette thèse. Ces jeux de données proviennent de compteurs électriques générant des courbes de charge prises à intervalles de temps différents. Le premier jeu de données dont nous disposons est pris à un intervalle de temps de 30 minutes et le second à un intervalle de temps de 10 minutes. Ces données sont celles disponibles aujourd'hui. Elles ne montrent pas un taux élevé de l'arrivée des éléments. Avec le déploiement envisagé à moyen terme de compteurs communicants chez les clients de EDF, les consommations d'énergie électrique pourront être télé-relevées à des pas de temps pouvant aller jusqu'à la seconde. Ce qui pourra fournir des jeux de données plus représentatifs de la problématique exposée dans cette thèse.

Chapitre 6

Echantillonnage temporel optimisé générique de flux de données distribués

Sommaire

6.1 Formulation du problème

Nous considérons un environnement distribué constitué de N *capteurs distants* et un *serveur central*. Les capteurs envoient une séquence de mesures temporelles en continu au serveur central qui se charge de répondre aux requêtes utilisateurs telles que la somme SUM ou la moyenne AVG portant sur un historique de tout ou partie des capteurs. Nous supposons que les communications entre le serveur central et les capteurs sont bi-directionnelles et qu'il n'existe pas de communication entre les capteurs. Cette architecture est représentative d'une large classe d'applications. Dans le cas particulier de la supervision de la consommation électrique, plusieurs millions de compteurs électriques communiquent leurs mesures de consommation à

un unique entrepôt de données en passant par des milliers de concentrateurs (voir 5). Ces derniers joueront le rôle de serveur central dans notre modèle. Plusieurs centaines de compteurs électriques sont connectés directement à un concentrateur. L'envoi des données se fait sous forme de flux, arrivant de façon continue à un pas de temps très fin (allant jusqu'à la seconde). Ceci génère une quantité volumineuse de données, et il devient coûteux de stocker et traiter toutes ces données. Afin de limiter cette quantité au concentrateur récoltant les données, nous imposons une limite de bande passante à ne pas dépasser et les données relevées à chaque pas de temps ne doivent pas excéder cette limite.

Le coût élevé de la collecte de données et la contrainte de la bande passante nous incitent à sélectionner un ensemble minimal de données échantillonnées tout en nous assurant de la représentativité de ces données pour nous approcher du flux de données initial. Nous présentons dans ce chapitre un outil générique de résumé en ligne de flux de données fortement distribuées. Ces algorithmes permettent de déterminer la « politique » de résumé sur une période temporelle t en prenant en compte des données de la période précédente $t-1$. En effet, nous estimons que les données provenant des capteurs ont des comportements semblables pendant deux périodes successives de longueur fixée. Nous démontrons l'efficacité de nos algorithmes en les appliquant aux jeux de données exposés au chapitre 5.

6.2 Résolution du problème

Sans perte de généralité, supposons que les flux de base sont réguliers. C'est-à-dire que les données sont générées régulièrement ou que des traitements de données manquantes sont appliqués aux flux de données. Nous découpons chaque flux en périodes (fenêtres temporelles) de même taille. Chaque fenêtre temporelle est constituée de p éléments. Nous supposons constant le nombre d'éléments d'une fenêtre car le flux est régulier. Par abus de langage, nous utiliserons le terme « courbe » pour qualifier chaque séquence temporelle issue d'un capteur. Le serveur central collecte un nombre de mesures compris entre m (paramètre qu'un utilisateur fixe) et p à partir de chaque capteur. Soit s le nombre d'éléments communicables au système central par les N capteurs sur la période t ($s < N * p$), c'est-à-dire le nombre de points que le système central peut accepter pendant la période (bande passante).

Plusieurs techniques exposées au chapitre 3 peuvent être utilisées pour résumer les flux de données provenant des capteurs. Nous ne détaillerons que trois techniques utilisées dans les expérimentations de validation de notre approche.

Le modèle de collecte de données doit préserver autant que possible les informations des flux de données en réduisant les erreurs de résumé. Plusieurs mesures d'erreurs peuvent être utilisées. Nous utiliserons la somme des erreurs quadratiques SSE (Sum of Square Errors), définie aussi comme le carré de la distance L_2 entre la courbe initiale $C = \{C(1), C(2), ..., C(p)\}$ et la courbe reconstruite à partir du

résumé $\hat{C} = \{\hat{C}(1), \hat{C}(2), ..., \hat{C}(p)\}$. SSE est calculée par l'expression suivante :

$$SSE(C, \hat{C}) = \sum_{i=1}^{p} (C(i) - \hat{C}(i))^2$$

Où p est le nombre d'éléments de la courbe C.

SSE est bien adapté à notre jeu de données réel, à savoir les courbes de consommation électrique [82]. Il est à noter que les mesures d'erreur ne peuvent pas être définies arbitrairement, certaines propriétés (non étudiées dans la présente thèse) sont nécessaires afin de déterminer la mesure à utiliser. Toutefois, nous avons expérimenté les distances L_∞ et L_1, nous discuterons dans la section 6.4 des conséquences de leur utilisation dans la résolution de notre problème.

Au cours de la période $t-1$, chaque capteur envoie les données résumées au serveur ainsi que les valeurs SSE obtenues en fonction du nombre de données qu'il aurait pu envoyer compris entre m et p valeurs. Ensuite, le serveur résout un problème d'optimisation linéaire afin de trouver le nombre optimal de données à collecter de chaque capteur pendant la période t. Nous supposons que le nombre de capteurs est suffisamment grand pour que les échanges soient uniformément distribués dans le temps et ainsi éviter la surcharge temporaire du serveur central.

6.2.1 Module côté capteur

Les trois méthodes de résumé sur lesquelles nous avons concentré nos travaux sont les suivantes : l'échantillonnage régulier, la segmentation de courbes et la compression par ondelettes. Nous détaillerons leur adaptation à notre problématique dans les sous-sections suivantes, le lecteur peut se référer au chapitre 3 pour plus de détails sur les ondelettes et la segmentation.

Echantillonnage régulier

Nous échantillonnons les courbes à un pas j inférieur à $m' = \lfloor \frac{p}{m} \rfloor$ ($\lfloor x \rfloor$ signifie partie entière de x). j correspond au 'saut' à effectuer entre deux points sélectionnés, par exemple $j = 2$ signifie que nous sélectionnons un point sur deux de la courbe. Cette méthode d'échantillonnage est qualifiée de « régulière » car les données prélevées sont équidistantes temporellement, nous effectuons un saut de j entre deux éléments échantillonnés. Le problème posé consiste à trouver des pas d'échantillonnage pour chaque capteur en respectant les contraintes sur la limite du serveur central et le nombre minimal de données à sélectionner par capteur. Les pas d'échantillonnage sont calculés à partir des points des courbes de la fenêtre temporelle $t-1$, et sont appliqués aux points de la période t. La somme des erreurs quadratiques dépend de la façon dont nous interpolons la courbe originale à partir des points échantillonnés . Nous avons choisi deux méthodes d'interpolation : « interpolation linéaire » et « interpolation constante ». La figure 6.1 montre un exemple de courbe de charge, et ses courbes échantillonnées reconstruites par interpolation constante et par interpolation linéaire.

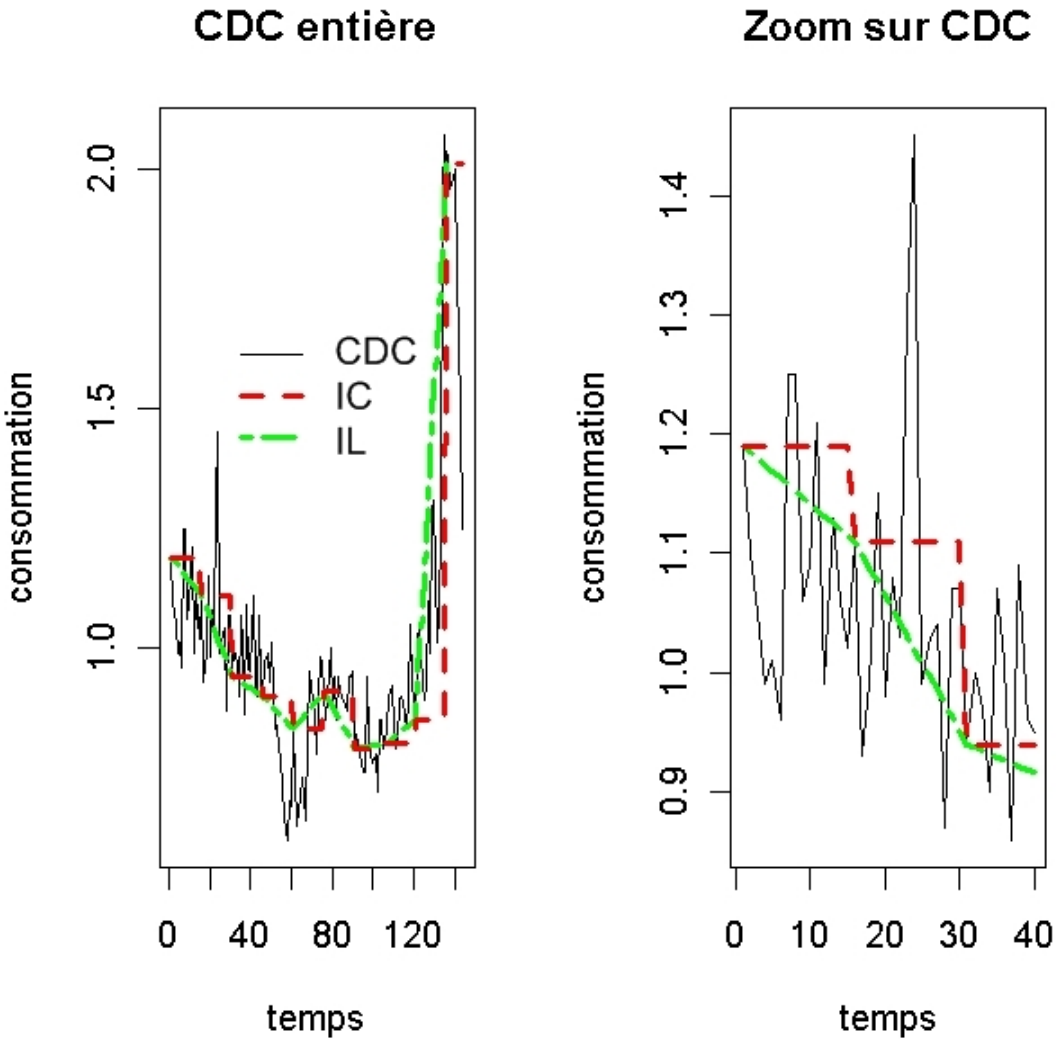

FIGURE 6.1 – Une Courbe initiale et ses courbes échantillonnées à pas=15 avec Interpolation constante (IC) et Interpolation linéaire (IL). La figure de droite correspond aux 40 premiers points de la Courbe. CDC : courbe initiale. IC : interpolation constante. IL : interpolation linéaire

Pour mettre à jour le vecteur des erreurs pour la période suivante, chaque capteur devrait envoyer m' SSEs. Cependant, il n'est pas toujours nécessaire d'envoyer la totalité des m' SSEs. En effet, si une courbe échantillonnée à un pas j, il suffit d'envoyer $m' - M_j$ SSEs, M_j étant le nombre de multiples de j qui sont inférieurs à m'. En effet, puisque le système central dispose des données échantillonnées, il peut donc calculer à partir de ces données et des SSEs envoyées par le capteur les SSEs manquant à la mise à jour complète de la matrice. A titre d'exemple, une courbe

échantillonnée à $j = 2$ et $m' = 20$ envoie à la fin de la période $m' - M_2 = 20 - 10 = 10$ SSEs, tel que $M_2 = card(\{2, 4, 6, ...20\}) = 10$.

Segmentation de courbes

Dans un objectif de compression, nous ne voulons conserver que les informations les plus pertinentes sur les courbes. Pour celà, nous utilisons la segmentation de courbe décrite au chapitre 3. Chaque courbe est découpée en épisodes et la moyenne des éléments est associée à chacun de ces épisodes. Le nombre de segments applicable à un capteur varie entre $\lfloor \frac{m}{2} \rfloor$ et p. En effet, pour reconstruire la courbe segmentée (en escalier) à partir des moyennes de chacun des épisodes, nous avons besoin aussi du nombre de points de chaque épisode. A titre d'exemple, si la courbe initiale C a été segmentée en k segments. Pour reconstruire la courbe segmentée, le serveur central a besoin des données $(\hat{C}(1), n_1), (\hat{C}(2), n_2), ..., (\hat{C}(k), n_k)$, $\hat{C}(l)$ étant la moyenne de l'épisode l constituée de n_l points. Donc, le compteur doit envoyer $2 * k$ données au serveur central qu'il faut prendre en compte pour ne pas dépasser le seuil s imposé par ce dernier. Par conséquent, on assigne à j (le nombre de segments à appliquer à un capteur) une valeur entière entre 1 (tous les éléments de la courbes sont sélectionnés) et m' (la courbe est segmentée de façon optimale en $\lfloor \frac{p}{2*m'} \rfloor$ segments). Comme pour l'échantillonnage régulier, chaque capteur effectue localement des segmentations avec différents nombres de segments, calcule les m' SSEs correspondants, indexés par une valeur allant de 1 à m' et les envoie au serveur central. Nous avons utilisé dans le cadre de ce travail la stratégie optimale employant la programmation dynamique [25] et qui a été exposée au chapitre 3.

Compression de courbes par ondelettes

Nous avons utilisé les ondelettes Haar car elles sont connues pour être bien adaptées à notre jeu de données réel (courbes de consommation électrique) [92]. La figure 6.2 montre un exemple de courbe de consommation électrique, et sa courbe compressée en escalier (reconstruite à partir de 6 coefficients ondelettes). L'estimation de la courbe à partir de sa version résumée est une combinaison linéaire d'un nombre j de vecteurs de bases compris dans l'ensemble $\{1, 2^1, 2^2, ...2^J\}$ ($j = 1$ signifie que la courbe est récupérée dans sa totalité). J est l'entier le plus élevé tel que $2^J \leq m'$. Dans le cas des ondelettes de Haar, le nombre d'éléments de la courbe doit être une puissance 2. Si ce n'est pas le cas, des zéros sont ajoutés à droite de la courbe afin de la compléter, on appelle cette technique le « Padding ». Ici aussi, chaque capteur effectue la décomposition en ondelettes à différents niveaux indexés par un j allant de 1 à $2^J \leq m'$ et envoie les SSEs correspondant au serveur central.

A la fin de chaque fenêtre temporelle, les capteurs calculent les SSEs correspondant aux méthodes de résumés définies (ici l'échantillonnage régulier, la segmentation et la compression par ondelettes) et envoient au serveur central pour chaque valeur de l'index j le minimum des erreurs entre ces méthodes.

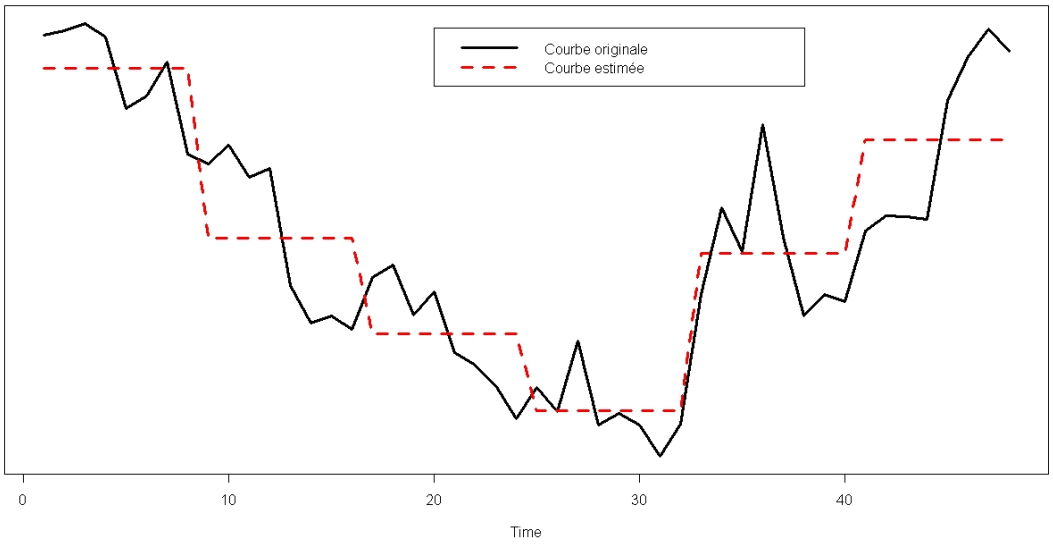

FIGURE 6.2 – Une courbe et sa courbe estimée à partir de 6 coefficients ondelettes

6.2.2 Module côté serveur

A la fin de chaque période temporelle, le serveur central reçoit de ses capteurs, différentes valeurs d'erreurs SSEs indexées par les niveaux de résumé correspondants j allant de 1 à m'. Ces SSEs sont stockées dans une matrice $W_{N*m'}$ de N lignes et m' colonnes (nous rappelons que N est le nombre de capteurs connectés au serveur central). Un élément w_{ij} de la matrice correspond à la somme des erreurs quadratiques obtenue si on collecte des données à partir du capteur i en utilisant un niveau de résumé j ($\left\lfloor \frac{p}{j} \right\rfloor$ éléments par échantillonnage régulier, $\left\lfloor \frac{p}{2*j} \right\rfloor$ éléments par segmentation et enfin $\frac{p}{2^{j-1}}$ éléments par ondelette).

La méthode la plus immédiate pour résumer les données est de partager équitablement la bande passante sur les différentes courbes, le niveau de résumé étant le même pour toutes les courbes. Cette solution respecte bien les contraintes posées mais ne minimise pas la somme des erreurs quadratiques entre les courbes initiales et les courbes résumées. De plus, les courbes présentant des fluctuations différentes, les niveaux de résumé « fixes » pourraient sur-représenter une courbe ou au contraire la sous-représenter. Pour résoudre le problème de minimisation des SSE, nous l'avons modélisé sous forme d'un problème d'optimisation linéaire que le serveur central se charge de résoudre à la fin de chaque période temporelle. Le problème s'énonce comme suit :

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

sous les contraintes :

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

Il s'agit d'un problème d'affectation de niveaux de résumé sur les différentes courbes en respectant les contraintes ci-dessus. Une variable X_{ij} à 1 signifie que le serveur central affecte le niveau de résumé j au capteur d'indice i. La deuxième contrainte $\sum_{j=1}^{m'} X_{ij} = 1$ impose une seule valeur de j par courbe (un seul pas d'échantillonnage, un seul nombre de segments ou un seul nombre de coefficients ondelettes par courbe). Enfin, la troisième contrainte signifie que le nombre de données à communiquer au système central ne doit pas dépasser le seuil imposé s. Une fois le problème d'optimisation résolu, les niveaux de résumés j sont envoyés à chaque capteur. Celui-ci utilise la méthode de résumé correspondant à la plus petite valeur de SSE pour ce niveau de résumé. A la fin de la période, les capteurs recalculent des SSEs pour mettre à jour la matrice des erreurs $W_{N*m'}$ comme il a déjà été expliqué ci-dessus. Ce processus continue tant que des flux de données arrivent en ligne.

Pour résoudre ce problème d'optimisation, nous utilisons la méthode du simplexe appliquée aux problèmes linéaires à variables réelles. Le simplexe est couplé avec la méthode Branch And Bound (Séparation-Evaluation en français) afin d'atteindre des variables entières. Le programme LP_Solve permet de résoudre des problèmes d'optimisation linéaires à variables réelles et/ou entières. Le lecteur peut se référer à [70; 90] pour des informations sur ces méthodes.

6.2.3 Approche générale

A la période $t = 0$, le serveur central n'a aucune connaissance des données des capteurs qui lui sont attachés. Celui-ci demande aux capteurs d'envoyer $\left\lfloor \frac{s}{N} \right\rfloor$ valeurs chacun tout au long de la première période, c'est une approche « fixe ». A l'expiration de la période, les capteurs calculent les erreurs correspondant aux trois méthodes de résumé (régulier, segmentation et compression par ondelettes) et envoient le minimum des erreurs entre ces trois méthodes au serveur central pour constituer la matrice des erreurs $W_{N*m'}$. Le serveur central applique le programme d'optimisation pour trouver les j correspondant aux pas d'échantillonnage dans le cas régulier, au double du nombre de segments dans le cas de la segmentation et au nombre de coefficients ondelettes dans le cas de la compression. Ensuite, il envoie le résultat de l'optimisation aux capteurs. Si le SSE en appliquant un pas d'échantillonnage j au capteur est inférieur au SSE en segmentant la courbe en $\left\lfloor \frac{p}{j \times 2} \right\rfloor$, ou en retenant $\frac{p}{2^{j-1}}$ coefficients ondelettes, alors le capteur décide d'appliquer un échantillonnage régulier et envoie les données au serveur central. Si le SSE par segmentation est le minimum, le capteur envoie les moyennes des segments accompagnés du nombre de points constituant le segment. Et enfin, si le SSE par compression est le minimum, le capteur envoie les coefficients d'ondelettes. A la fin de la deuxième période $t = 1$, les capteurs envoient de la même façon que précédemment les SSEs pour mettre à jour

la matrice des erreurs. Ce processus continue tant que les flux de données arrivent en ligne. Le schéma 6.3 résume le processus d'échange entre le serveur central et les capteurs.

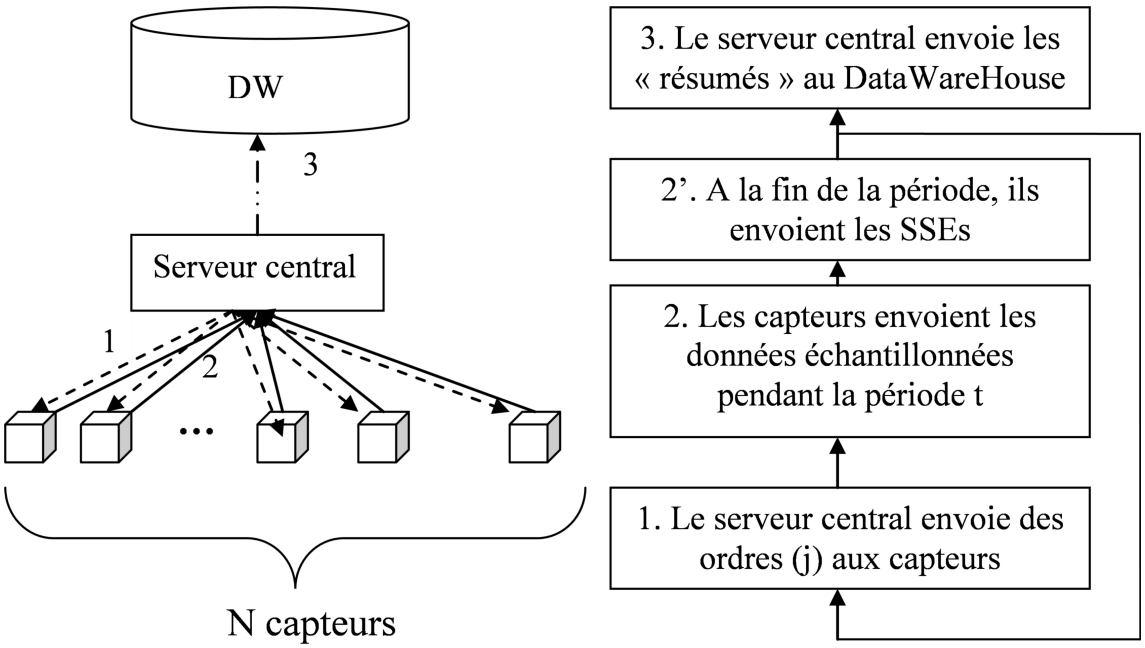

FIGURE 6.3 – Processus d'échange entre le serveur central et les capteurs

Voici les algorithmes du processus d'échange, incluant :
- L'algorithme 2 'ModuleCapteur' : permet de calculer les SSEs et de les envoyer au serveur central ;
- L'algorithme 3 'ModuleServeur' : met à jour la matrice des erreurs puis optimise les niveaux de résumé à appliquer à chaque capteur pour la période suivante ;
- L'algorithme 4 'RésuméCapteur' : applique le niveau de résumé affecté en choisissant la méthode la plus adaptée, puis fait appel à l'algorithme 'ModuleCapteur' afin de calculer les SSEs pour la période courante.

6.3 Validation expérimentale

L'approche décrite dans ce chapitre a été évaluée de façon empirique sur les jeux de données décrits au chapitre 5, et a été comparée à une approche non adaptative dite approche *fixe*. Dans cette approche, les trois méthodes de résumé sont utilisées mais le même nombre d'éléments sélectionnés est distribué équitablement entre les capteurs en fonction du paramètre s.

Nous avons réalisé nos premières expérimentations sur le second jeu de données présenté au chapitre 5 constitué de 1000 capteurs chacun produisant une mesure

Algorithm 2 Algorithme ModuleCapteur

Require: t - Période
Require: C - Courbe de la période t (p élements)
Require: m - nombre minimum de données à envoyer
Ensure: vecteur de m' SSEs
  $m' = \lfloor \frac{p}{m} \rfloor$
  for i from 1 to m' **do**
    for j in MéthodesRésumé **do**
      SSE<-Ajouter(CalculSSE(C,i,j))
    end for
    SSEs<-Ajouter(min(SSE))
  end for
  Envoyer(SSEs)

Algorithm 3 Algorithme ModuleServeur

Require: $W_{N*m'}$
Require: s - seuil (nombre de données maximum à accepter)
Ensure: Niv - N niveaux de résumé à appliquer aux N capteurs
  Niv<-optimise($W_{N*m'}$,s)
  for i de 1 à N **do**
    Envoyer(Niv[i],i) - Envoie le niveau de résumé pour le capteur d'indice i
  end for

Algorithm 4 Algorithme RésuméCapteur

Require: t - Période
Require: C - Courbe de la période t
Require: m - nombre minimum de données à envoyer
Require: Niv - Le niveau de résumé affecté par le serveur
Ensure: Résumé
  for j in MéthodesRésumé **do**
    SSE<-Ajouter(CalculSSE(CDC,Niv,j))
  end for
  MéthodeRésumé <- min(SSE)
  Résumé<-appliquer(MéthodeRésumé,C,Niv)
  Envoyer(Résumé)
  ModuleCapteur(C,t,m)

toutes les 10 minutes. Nous n'avons utilisé que les données d'une seule journée (144 mesures par compteur). Ce jeu de données a servi à évaluer l'efficacité de notre approche dans un cas appelé « stationnaire » : les niveaux de résumé optimisés sont appliqués aux courbes de la même période que celle qui a servi à les choisir. Ces expérimentations sont qualifiées par « fenêtre stationnaire » ci-dessous.

Les secondes expérimentations ont été effectué sur le premier jeu de données correspondant à 140 compteurs électriques chacun produisant une mesure toutes les 30 minutes et ceci pendant 365 jours (un an) (cf. chapitre 5). Ce jeu de données est utilisé afin d'évaluer l'efficacité de l'approche dans le cas d'une consommation non stationnaire pour chaque client : les niveaux de résumé optimisés sont appliqués aux courbes de la période suivant celle utilisée pour choisir ces niveaux. Ces expérimentations sont qualifiées par « fenêtre sautante » ci-dessous.

6.3.1 Fenêtre stationnaire

Nous avons considéré $m = 7$ ($m' = \lfloor \frac{144}{7} \rfloor = 20$) ce qui signifie qu'une courbe échantillonnée sera constituée d'au moins 7 valeurs si elle est construite de façon régulière, de 3 valeurs si elle est construite avec la segmentation et de 9 avec la compression par ondelettes. Nous avons testé plusieurs valeurs de bande passante (seuil s) et nous avons dressé un tableau récapitulatif des résultats obtenus (tableau 6.1). Chaque colonne du tableau correspond à une valeur du seuil s. $s = p \times N = 144 * 1000$ signifie que toutes les mesures des courbes ont été collectées. Les seuils expérimentés sont égaux à $p \times N = 144000$ divisé par une valeur de l'ensemble $\{1, 2, 3, 5, 7, 10, 15\}$.

Nous avons calculé la somme des erreurs quadratiques SSE pour chacune des méthodes. Nous avons aussi calculé l'écart-type pour mesurer la dispersion du nombre d'éléments affecté par optimisation autour de la moyenne. Celle-ci est égale à $\frac{s}{1000}$ ($N = 1000$).

Les résultats du tableau 6.1 font apparaître que l'optimisation permet de diminuer significativement la somme des erreurs quadratiques pour toutes les techniques de résumé utilisées par rapport à l'approche 'fixe'. Par exemple, dans le cas de l'échantillonnage régulier avec interpolation linéaire, l'optimisation a permis de diviser par 4,5 l'erreur quadratique pour un seuil de $s = \frac{p*N}{2}$ par rapport à l'échantillonnage régulier 'fixe'. Le coût de transmission des SSEs n'a pas été pris en compte dans les présentes expérimentations. Ceci a un impact sur la comparaison entre les deux approches 'fixe' et 'optimisée', puisque la transmission des SSEs n'est pas utile dans le cas de l'approche 'fixe'. Toutefois, le nombre des valeurs SSE à transmettre est limité et peut être considéré négligeable comparé au nombre p d'éléments dans une fenêtre temporelle.

Nous observons des valeurs élevées des écarts-types surtout pour les valeurs élevées du seuil s. Les niveaux de résumé sont largement écartés de la moyenne. La figure 6.4 est un exemple de distribution des pas d'échantillonnage (interpolation linéaire) pour un seuil de $\frac{p*N}{5}$. Nous remarquons que les pas d'échantillonnage par optimisation sont répartis sur les différentes valeurs des pas avec une concentration au niveau des limites (*j=1* et *j=20*), alors qu'en échantillonnage 'fixe' le pas d'échan-

Seuil s	$p*N$	$\frac{p*N}{2}$	$\frac{p*N}{3}$	$\frac{p*N}{5}$	$\frac{p*N}{7}$	$\frac{p*N}{10}$	$\frac{p*N}{15}$
SSE opt Samp CI	0	696.36	1571.79	3037.02	4265.46	5839.95	8767.05
SSE opt Samp LI	0	434.35	996.34	1969.67	2781.54	3883.9	5676.6
SSE opt seg	0	161.62	385.13	874.1	1346.58	2027.17	3432.16
SSE opt wav	0	381.15	916.22	1881.66	2724.08	3888.6	6118.8
SSE fixe CI	0	2770.8	4329	8040	9313	13157	16048
SSE fixe LI	0	1956	2821	4245	5501	7024	9087
SSE fixe seg	0	505.8	860.8	1535	2042.6	2761.5	4677.1
SSE fixe wav	0	1385.32	2725.35	4766.09	4766.09	7179.5	7179.5
avg data/meter	144	72	48	28.8	20.57	14.4	9.6
std data CI	0	50.3	44.8	32.05	24.48	15.62	4.4
std data LI	0	51	45.45	33.6	24.7	16.5	4.9
std data seg	0	43	35.73	25.32	17.64	10.6	3.9
std data wav	0	48.9	42	30	21.8	13.5	3

TABLE 6.1 – Tableau récapilutaif. SSE : Sum of Square Error. opt : niveaux de résumé obtenus par optimisation. avg : Moyenne and std : Ecart-type. Samp CI : Echantillonnage régulier avec interpolation constante, Samp LI : Echantillonnage régulier avec interpolation linéaire, seg : Segmentation et wav : Compression par ondelettes.

tillonnage est *j=5*. Ce fait permet de confirmer l'importance de sélectionner les pas d'échantillonnage par optimisation.

Au vu des résultats du tableau 6.1, nous remarquons que la segmentation permet en général sur ce jeu de données de mieux minimiser la somme des erreurs quadratiques par rapport à l'échantillonnage régulier et à la compression par ondelettes. La segmentation donne des résultats meilleurs malgré le nombre inférieur de données des courbes récupérées par période de temps. A titre d'exemple, pour un seuil de $\frac{p*N}{15}$, la segmentation a divisé la somme des erreurs quadratiques par 2,55 par rapport à l'échantillonnage régulier avec interpolation constante, par 1,7 par rapport à l'échantillonnage régulier avec interpolation linéaire et par 1.8 par rapport à la compression par ondelettes.

Quelque soit le taux de compression, les méthodes de résumé par optimisation donnent de meilleurs résultats que les méthodes à niveau de résumé fixe. Néanmoins, pour des taux de compression très élevés, nous remarquons que quelques méthodes fixes réduisent les erreurs de résumé par rapport aux méthodes optimisées. Par exemple, pour un seuil de $\frac{p*N}{15}$, il est plus intéressant d'appliquer la segmentation fixe aux capteurs, car elle est meilleure que l'échantillonnage régulier optimisé et la compression par ondelettes optimisée, et que le gain n'est pas très significatif par rapport à la segmentation optimisée. D'autant plus, que nous gagnons en temps et coût de traitement puisque nous éliminons la phase d'optimisation.

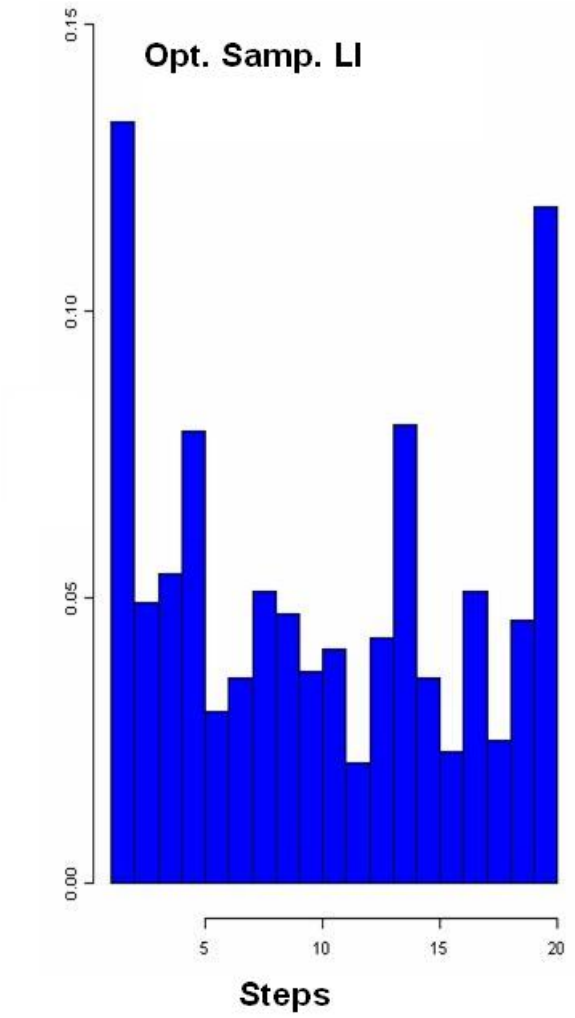

FIGURE 6.4 – distribution des pas d'échantillonnage

6.3.2 Fenêtre stationnaire avec sélection de méthode

Nous avons expérimenté le cas où les compteurs choisissent eux-mêmes la méthode de résumé à adopter en fonction du j affecté afin de réduire la somme des erreurs quadratiques le plus possible. Nous observons que le SSE est encore minimisé grâce à la sélection automatique (cf. la ligne 'SSE opt' du tableau 6.2) même si la différence n'est pas très significative par rapport au résumé par segmentation. Nous n'avons pas pris en compte le gain en coût de traitement, mais il est évident qu'il en existe un grâce à la sélection automatique de méthode, puisque nous réduisons le nombre de courbes que nous segmentons (complexité en $O(kp^2)$) au profit de l'échantillonnage régulier (complexité en $O(p)$).

Le tableau 6.2 montre les résultats correspondant au pourcentage d'utilisation de chaque méthode de résumé. 'Courbe entière' dans le tableau correspond au pourcentage des capteurs pour lesquels un niveau de résumé égal à 1 ($j = 1$) est affecté. Ce qui signifie que nous récupérons la totalité des mesures des courbes. Au départ,

nous avons expérimenté l'ensemble des trois méthodes décrites précédemment. Nous remarquons que la segmentation est de loin la méthode la plus utilisée (plus de 74%) et ceci pour toute valeur de seuil, suivie par l'échantillonnage régulier.

Seuil s	$p*N$	$\frac{p*N}{2}$	$\frac{p*N}{3}$	$\frac{p*N}{5}$	$\frac{p*N}{7}$	$\frac{p*N}{10}$	$\frac{p*N}{15}$
% Echant. régulier	0	2	3	7	10	11	14
% Segmentation	0	74	87	90	89	88	85
%Ondelettes	0	0	0	1	1	1	1
%Courbe entière	100	24	10	4	1	0	0
SSE opt	0	139.29	346.05	801.91	1240.62	1859.6	3026.24

TABLE 6.2 – Taux d'utilisation de chaque méthode. Courbe entière signifie que tous les points ont été collectés (pas de méthode de résumé appliquée).

Nous avons ensuite effectué des expérimentations sans la méthode par segmentation, les résultats sont présentés dans le tableau 6.3. Comme nous le remarquons, les courbes utilisent presque équitablement les deux méthodes de résumé pour les trois premières valeurs de seuil. Pour les valeurs élevées de s, l'échantillonnage régulier est la méthode la plus utilisée par les capteurs. Nous avons aussi remarqué que les courbes avec une forme constante choisissent d'utiliser l'échantillonnage régulier avec interpolation constante plutot que l'interpolation linéaire quand un pas d'échantillonnage élevé leur est affecté.

Seuil s	$p*N$	$\frac{p*N}{2}$	$\frac{p*N}{3}$	$\frac{p*N}{5}$	$\frac{p*N}{7}$	$\frac{p*N}{10}$	$\frac{p*N}{15}$
% Echant. régulier	0	28	40	54	58	64	73
%Ondelettes	0	44	46	41	40	35	27
%Courbe entière	100	28	14	5	2	1	0
SSE opt	0	354.23	832.05	1679.21	2396.15	3352.41	5005.19

TABLE 6.3 – Taux d'utilisation de chaque méthode (sans la segmentation). Courbe entière signifie que tous les points ont été collectés (pas de méthode de résumé appliquée).

6.3.3 Fenêtre sautante

Les expérimentations précédentes évaluent l'erreur commise sur une période t à partir d'une optimisation effectuée sur cette même période. L'approche globale proposée consiste à appliquer sur une période t le résultat d'une optimisation réalisée sur la période $t-1$. Nous présentons ici les résultats des expérimentations réalisées sur les 140 courbes de charge disponibles sur une période d'une année.

Optimisation grâce à j-1

Le tableau 6.4 présente le pourcentage de périodes (fenêtres temporelles) pour lesquelles le SSE en utilisant le résumé 'optimisé' est inférieur au SSE du résumé 'fixe'. La durée utilisée pour la fenêtre temporelle est d'une journée. La phase d'optimisation utilise les données de la journée précédente et les niveaux de résumé sont appliqués à la journée courante, ceci tout au long de l'année. Les seuils expérimentés sont égaux au nombre total des éléments $p * N = 48 * 140 = 6720$ divisé par chacune des valeurs de l'ensemble $\{1, 2, 3, 5, 7, 10, 15\}$.

Seuil s	$p*N$	$\frac{p*N}{2}$	$\frac{p*N}{3}$	$\frac{p*N}{5}$	$\frac{p*N}{7}$	$\frac{p*N}{10}$	$\frac{p*N}{15}$
%j< 10% SSE LI opt-1	-	84%	84%	83%	85%	85%	55%
%j< 20% SSE LI opt-1	-	81%	80%	75%	84%	83%	7%
%j < 10% SSE CI opt-1	-	80%	81%	83%	85%	88%	96%
%j < 20% SSE CI opt-1	-	75%	73%	80%	83%	86%	75%
%j < 10% SSE seg opt-1	-	82%	81%	84%	84%	84%	84%
%j < 20% SSE seg opt-1	-	81%	78%	84%	84%	84%	84%
%j < 10% SSE wav opt-1	-	84%	84%	88%	86%	100%	80%
%j < 20% SSE wav opt-1	-	83%	84%	87%	85%	95%	5%

TABLE 6.4 – Pourcentage des jours pour lesquels la méthode par optimisation est meilleure que la méthode fixe. %j< X% SSE ($X \in \{10, 20\}$ correspond au pourcentage de jours dont le SSE par optimisation est inférieur de X% au SSE obtenu par méthode de résumé fixe. opt-1 : Optimisation utilisant les données du jour $j-1$ et le résumé effectué sur le jour j.

Les résultats du tableau 6.4 montrent que la méthode reste performante. En effet, l'optimisation des niveaux de résumé sur le flux permet de diminuer d'au moins 10% par rapport à la méthode 'fixe' les erreurs de résumé de plus de la moitié des jours de l'année, pour toutes les valeurs de seuil expérimentées. Nous observons que pour le seuil $s = \frac{p*N}{15}$, le nombre de jours pour lesquels l'optimisation donne de meilleurs résultats (au moins de 20%) est très inférieur par rapport aux autres seuils expérimentés pour l'échantillonnage régulier avec interpolation linéaire et pour la compression par ondelettes. Ceci est dû au nombre de points récupérés pour ce seuil qui est très bas (en moyenne 3 éléments par capteur). Ainsi, le résumé des flux que ce soit optimisé ou fixe n'est pas très performant. Ceci rejoint notre observation au vu du tableau 6.1 de la sous-section 6.3.1, à savoir que pour un grand taux de compression, il est préférable d'appliquer une méthode de résumé fixe, qui est moins coûteuse que si on avait appliqué le module d'optimisation pour affecter les niveaux de résumé.

La figure 6.5 représente des boxplots qui fournissent un résumé visuel des aspects de la distribution des SSEs pendant une année en utilisant un échantillonnage régulier optimisé et fixe. Nous observons que pour un seuil de $s = \frac{p*N}{5} = 1344$, le SSE en

utilisant l'optimisation est meilleur que le SSE avec la méthode fixe. Dans le boxplot de la méthode optimisée, nous pouvons voir de nombreuses valeurs aberrantes qui impliquent une haute valeur moyenne. En effet, la moyenne du SSE en utilisant l'optimisation est de $avg_{opt-1} = 554185.1$, alors que la moyenne du SSE en utilisant l'échantillonnage fixe est $avg_{fix} = 485025.7$.

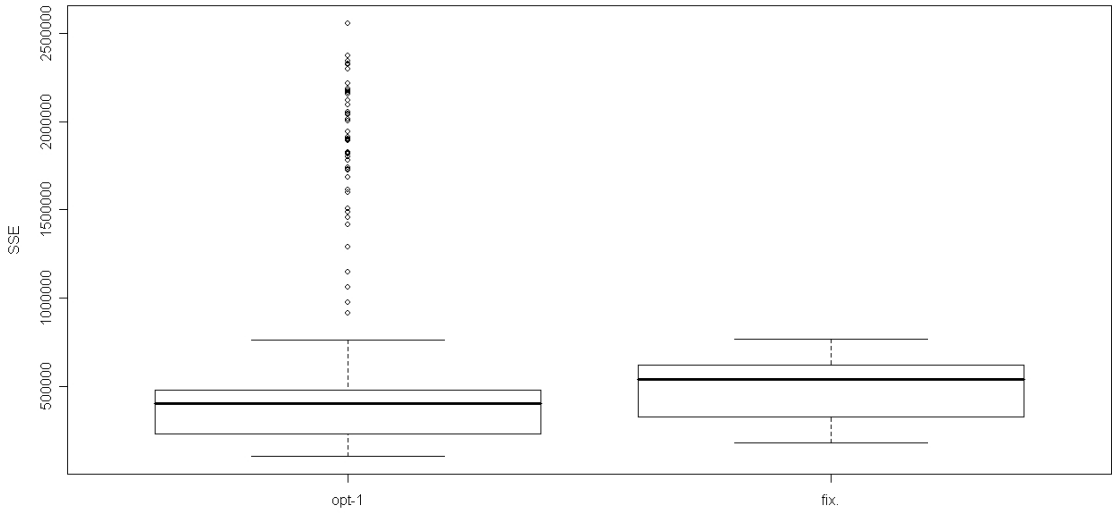

FIGURE 6.5 – La distribution des SSE (s=1344) opt-1 : Optimisation utilisant les données de j-1 et résumé des données de j. fix. : Méthode de résumé fixe.

Optimisation grâce à j-7

Les expérimentations précédentes ont été effectuées sur les données du jour $j-1$ pour résumer les données du jour j. La figure 6.6 montre l'évolution des SSEs durant un mois (30 jours) pour un seuil de $s = \frac{p*N}{5}$. Une périodicité des erreurs apparaît sur la courbe correspondant à l'évolution des SSEs en appliquant les fréquences d'échantillonnage obtenues par optimisation pendant la journée précédente (courbe opt-1). Ceci est dû au cycle hebdomadaire de la consommation d'électricité (5 jours ouvrables avec une consommation globalement stable et le week end où la consommation change de niveau). Dans la méthode, nous déterminons les niveaux de résumé pour les journées du lundi en nous basant sur les données du dimanche et celles du samedi par les données du vendredi. Nous avons expérimenté le cas où la phase de résumé sur les données de la journée courante j se fait grâce aux niveaux de résumé calculées à partir de la courbe de la journée $j-7$. Au vu de la courbe opt-7 de la figure 6.6 nous remarquons que cette méthode permet de 'lisser' la courbe de l'évolution des erreurs d'échantillonnage et ainsi diminuer les erreurs dûes à la périodicité hebdomadaire de la consommation électrique.

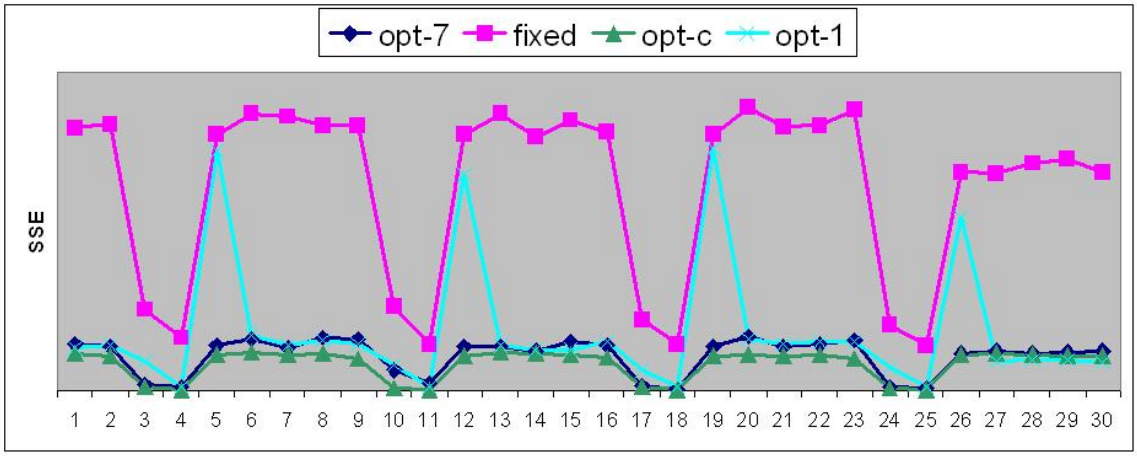

FIGURE 6.6 – Evolution des SSEs pendant 30 jours. opt-7 : Optimisation utilisant les données de j-7 et résumé des données de j. opt-1 : Optimisation utilisant les données de j-1 et résumé des données de j. opt-c : Optimisation et résumé des données du jour j (jour courant). fixed : Méthode de résumé fixe.

La figure 6.7 représente un boxplot de la distribution des SSEs en utilisant les données $j-7$ pour résumer par échantillonnage régulier les données de la journée courante j. Nous observons que les valeurs atypiques (ou outliers) ont été réduits et nous pouvons le vérifier grâce au calcul des valeurs de la moyenne des SSEs. En effet, la moyenne des SSEs en utilisant l'optimisation est de $avg_{opt-7} = 408079.8$ et est inférieure à la moyenne de la méthode 'fixe' qui est de $avg_{fix} = 485294.5$.

Le tableau 6.5 montre que les méthodes de résumé avec optimisation en utilisant $j-7$ réduisent toujours au moins de 10% le SSE comparé aux méthodes fixes pour plus de 70% des jours de l'année et ceci pour toute valeur de seuil utilisée.

Seuil s	$p*N$	$\frac{p*N}{2}$	$\frac{p*N}{3}$	$\frac{p*N}{5}$	$\frac{p*N}{7}$	$\frac{p*N}{10}$	$\frac{p*N}{15}$
%j < 10% SSE IL opt-7	0	83%	83%	84%	92%	94%	89%
%j < 10% SSE esc opt-7	0	79%	83%	92%	98%	98%	98%
%j < 10% SSE seg opt-7	0	77%	75%	88%	89%	97%	93%
%d < 10% SSE wav opt-7	0	87%	96%	98%	98%	100%	94%

TABLE 6.5 – Pourcentage des jours pour lesquels la méthode par optimisation est meilleure que la méthode fixe. %j< 10% SSE correspond au pourcentage de jours dont le SSE par optimisation est inférieur de 10% au SSE obtenu par méthode de résumé fixe. opt-7 : Optimisation utilisant les données du jour $j-7$ et le résumé effectué sur le jour j.

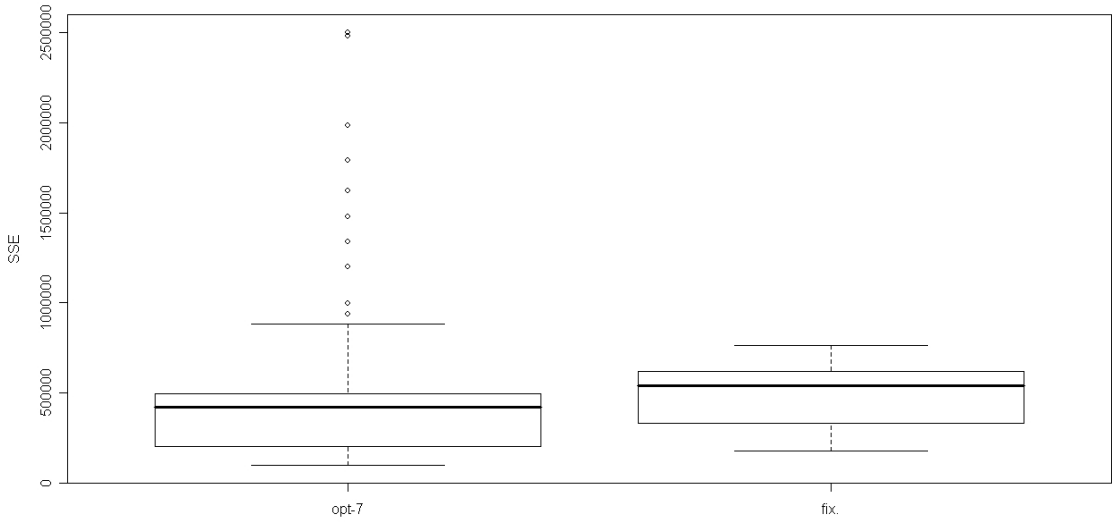

FIGURE 6.7 – La distribution des SSE (s=1344) opt-7 : Optimisation utilisant les données de j-7 et résumé des données de j. fix. : Méthode de résumé fixe.

Optimisation grâce à s-1

Nous avons également expérimenté une durée de la période t égale à la semaine, en suivant l'évolution des SSEs durant les 52 semaines de l'année. Dans ce cas aussi, l'approche permet de diminuer les erreurs d'échantillonnage par rapport à l'échantillonnage fixe, et ceci pour plus de 86% des semaines de l'année pour toutes les méthodes de résumé expérimentées (voir le tableau 6.6). La figure 6.8 montre l'évolution des erreurs durant une période d'une année et utilisant une fenêtre temporelle d'une semaine avec un seuil $s = \frac{p*N}{5}$. A l'inverse de l'optimisation grâce à $j-1$, nous n'observons pas de cycle dans l'évolution des erreurs. Les SSEs en utilisant l'approche optimisée sont clairement réduits par rapport à l'approche 'fixe' et durant toute l'année.

Les boxplots de la figure 6.9 montrent la distribution du SSE en utilisant les approches 'fixe' et optimisée. Les Boxplots confirment l'utilité de l'optimisation par rapport à l'approche 'fixe'. Nous observons qu'il y a moins d'éléments atypiques dans le cas où on choisit une fenêtre temporelle d'une semaine $(s-1)$ que dans le cas de la période d'une journée $(j-1)$. Toutefois, il est plus intéressant d'utiliser une période temporelle d'une journée qu'une période d'une semaine, car les erreurs de résumé augmentent considérablement dans le second cas.

6.3.4 Influence du paramètre m

Nous avons voulu connaître l'influence du paramètre m qui est le nombre minimal des données à prélever de chaque capteur. Pour cela, nous avons effectué des

Seuil s	$p*N$	$\frac{p*N}{2}$	$\frac{p*N}{3}$	$\frac{p*N}{5}$	$\frac{p*N}{7}$	$\frac{p*N}{10}$	$\frac{p*N}{15}$
%sem. $<$ 10% SSE IL opt-1	0	100%	100%	100%	100%	100%	100%
%sem. $<$ 10% SSE esc opt-1	0	98%	100%	100%	100%	100%	100%
%sem. $<$ 10% SSE seg opt-1	0	94%	94%	92%	90%	90%	86%
%sem. $<$ 10% SSE wav opt-1	0	100%	100%	100%	100%	100%	100%

TABLE 6.6 – Pourcentage des semaines (sem.) pour lesquelles la méthode par optimisation est meilleure que la méthode fixe. %sem $<$ 10% SSE correspond au pourcentage de semaines dont le SSE par optimisation est inférieur de 10% au SSE obtenu par méthode de résumé fixe.

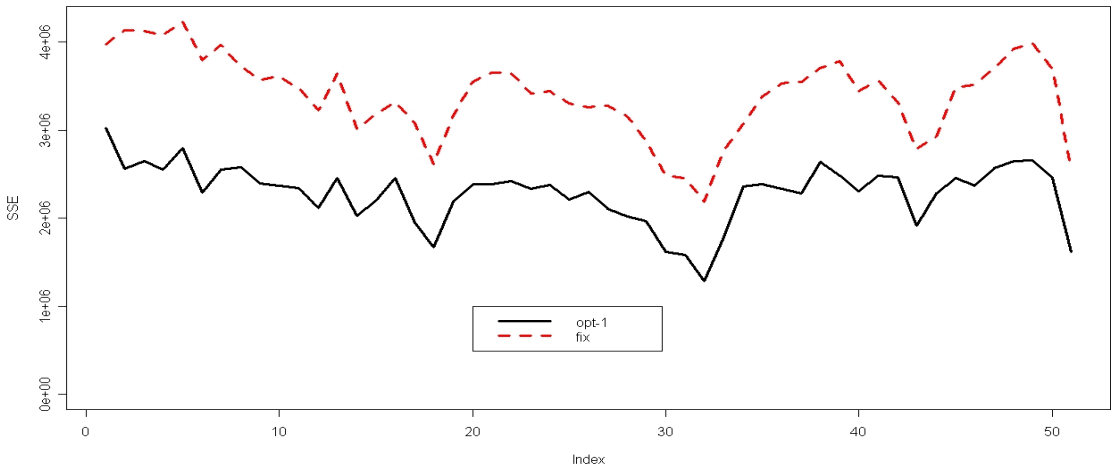

FIGURE 6.8 – Evolution des SSEs pendant une année. opt-1 : Optimisation utilisant les données de la semaine s-1 et résumé des données de s. fix. : Méthode de résumé fixe.

expérimentations avec différentes valeurs de m. La figure 6.10 montre l'évolution des erreurs en fonction de $m' = \lfloor \frac{p}{m} \rfloor$, ici $p = 144$ le nombre de points de la courbe par période t (une journée), ainsi que la distribution des niveaux de résumés en fonction de m'. Nous disposons de $N = 1000$ courbes. Le seuil utilisé est $s = \frac{p*N}{3} = 48000$. Nous observons une faible diminution des erreurs avec l'augmentation de m' ce qui donne pour les capteurs plus de choix de niveaux de résumé. Ainsi, si un capteur génère des données constantes, plus le niveau de résumé qu'il choisit est grand, plus il restera de la place à un capteur avec des fluctuations dans ses données. Toutefois la différence n'est pas très significative. Quant à la distribution des niveaux, elle est quasiment identique pour les trois valeurs de m' (figure 6.10 (a)), ceci est confirmé par la figure 6.11. On peut donc considérer un m tel qu'on arrive à résoudre le problème d'optimisation des niveaux de résumé dans un temps raisonnable. Dans nos expérimentations, nous avons utilisé $m = 7$ (donc $m' = 20$) avec un nombre de cap-

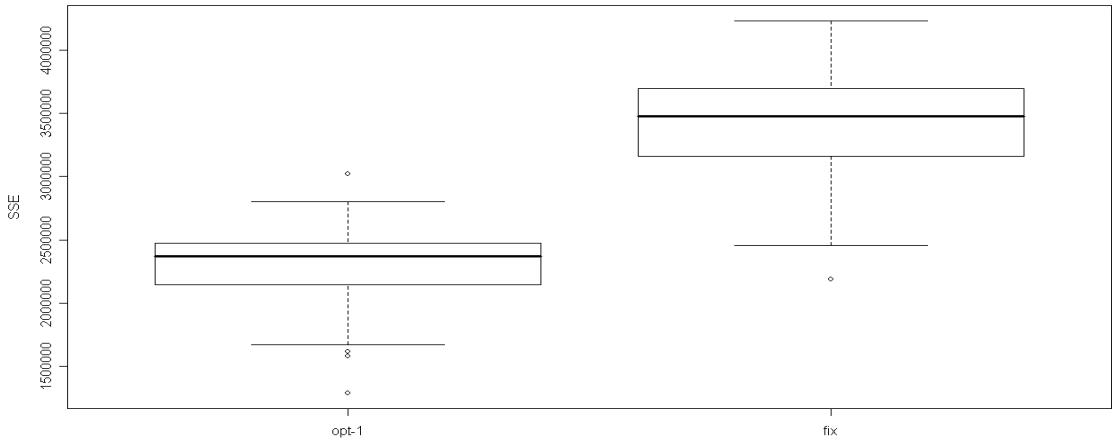

FIGURE 6.9 – La distribution des SSE (s=$\frac{p*N}{5}$) opt-1 : Optimisation utilisant les données de la semaine s-7 et résumé des données de s. fix. : Méthode de résumé fixe.

teurs égal à 1000. Avec ces paramètres, la résolution du programme d'optimisation linéaire est accomplie en moins d'une minute.

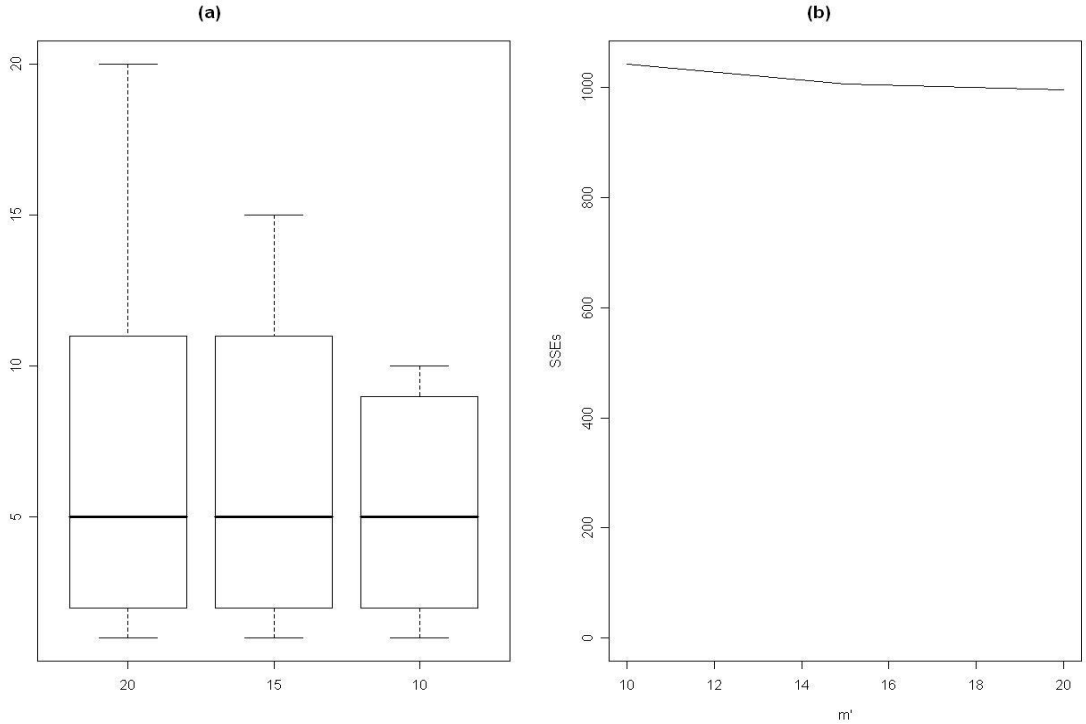

FIGURE 6.10 – Influence de m' sur les erreurs et la distribution des niveaux de résumé

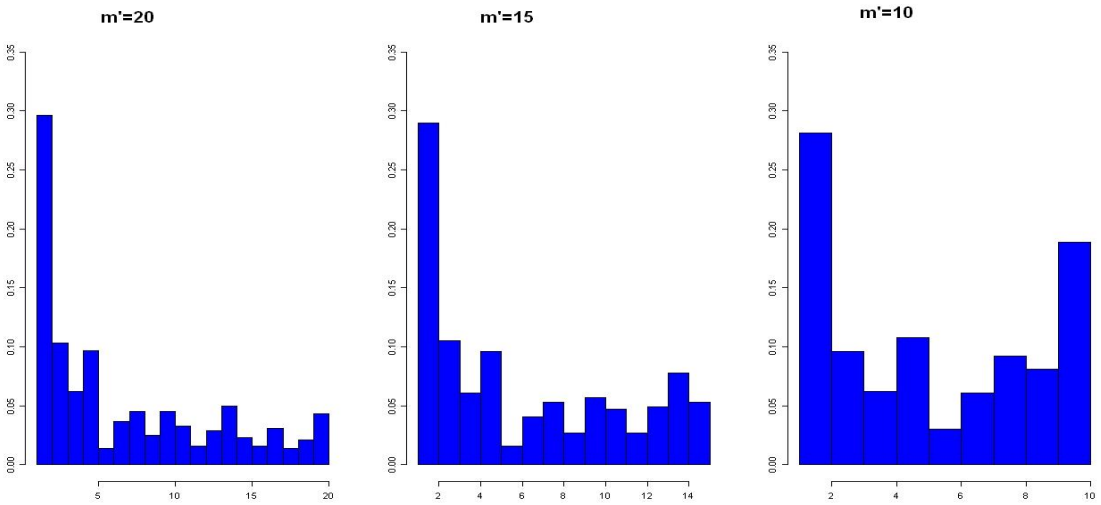

FIGURE 6.11 – Distribution des niveaux de résumé en variant m'

6.3.5 Encore plus d'expérimentations

Nous avons effectué des expérimentations supplémentaires sur le second jeu de données (cf. chapitre 5) constitué de 1000 courbes de charge correspondant à une période d'un an et mesurées à des intervalles de temps de 10 minutes. Nous avons testé deux tailles de fenêtres temporelles (journée et semaine) et nous avons suivi l'évolution des erreurs dans le temps. Nous avons utilisé l'échantillonnage régulier comme méthode de résumé avec les deux techniques d'interpolation (linéaire et constante). Nous nous sommes restreint à une seule méthode de résumé car les expérimentations mettent beaucoup de temps à traiter les données d'une année.

Seuil s	$p*N$	$\frac{p*N}{2}$	$\frac{p*N}{3}$	$\frac{p*N}{5}$	$\frac{p*N}{7}$	$\frac{p*N}{10}$	$\frac{p*N}{15}$
%j< 10% SSE LI opt-1	-	83%	84%	85%	85%	88%	99%
%j < 10% SSE CI opt-1	-	82%	84%	85%	84%	85%	90%
%j< 10% SSE LI opt-7	-	95%	96%	96%	95%	97%	99%
%j < 10% SSE CI opt-7	-	93%	96%	96%	96%	97%	99%

TABLE 6.7 – Pourcentage des jours pour lesquels la méthode par optimisation est meilleure que la méthode fixe. %j< 10% SSE correspond au pourcentage de jours dont le SSE par optimisation est inférieur de 10% au SSE obtenu par méthode de résumé fixe. opt-1 : Optimisation utilisant les données du jour $j-1$ et le résumé effectué sur le jour j.opt-7 : Optimisation utilisant les données du jour $j-7$ et le résumé effectué sur le jour j

Les tableaux 6.7 et 6.8 montrent que notre outil donne des résultats très satisfaisants pour ce grand jeu de données, que ce soit en utilisant une période temporelle

Seuil s	$p*N$	$\frac{p*N}{2}$	$\frac{p*N}{3}$	$\frac{p*N}{5}$	$\frac{p*N}{7}$	$\frac{p*N}{10}$	$\frac{p*N}{15}$
%s< 10% SSE LI opt-1	-	98%	98%	98%	98%	100%	100%
%s < 10% SSE CI opt-1	-	98%	98%	100%	100%	100%	100%

TABLE 6.8 – Pourcentage des semaines pour lesquelles la méthode par optimisation est meilleure que la méthode fixe. %s< 10% SSE correspond au pourcentage de semaines dont le SSE par optimisation est inférieur de 10% au SSE obtenu par méthode de résumé fixe.

d'une journée (avec une optimisation effectuée sur la journée $j-1$ ou sur $j-7$ pour échantillonner la journée courante j) ou pour une période temporelle d'une semaine. En effet, l'affectation des fréquences d'échantillonnage par optimisation réduit les erreurs pour plus de 80% des jours de l'année pour toute valeur de seuil expérimentée, et pour plus de 98% des semaines de l'année.

6.4 Autres mesures d'erreur

Afin d'affecter les niveaux de résumé aux courbes en minimisant les erreurs, nous avons modélisé le problème sous forme de programme d'optimisation linéaire avec une fonction objectif qui consiste en une somme des erreurs L^2 pour l'ensemble des courbes. Nous rappelons l'expression de la fonction objectif :

$$\sum_{i=1}^{N}\sum_{j=1}^{m'}(W_{ij} \times X_{ij})$$

Toutefois, l'un des problèmes de la norme L^2 peut être l'existence d'erreurs ponctuelles qui peuvent être très importantes pour certaines courbes. Afin de prendre en compte ces informations, on utilise généralement une erreur relative, et la fonction objectif s'exprime par :

$$\sum_{i=1}^{N}\frac{(\sum_{j=1}^{m'}(W_{ij} \times X_{ij}))}{\frac{1}{m'}W_{ij}}$$

Cette écriture ne modifie pas la résolution du programme d'optimisation linéaire exposée précédemment.

Une deuxième façon de prendre en compte les erreurs ponctuelles est d'utiliser une norme L_∞, qui consiste à considérer le maximum des erreurs entre les courbes. Ceci implique des changements dans la résolution du problème d'optimisation que nous allons expliciter dans ce qui suit.

Le problème à résoudre s'écrit sous la forme suivante :

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

sous les contraintes :

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

Ce problème est appelé dans la littérature *Problème de Chebychev* ou *Problème minimax*. Il peut être transformé en problème d'optimisation linéaire grâce à l'introduction d'une nouvelle variable que nous allons appeler Y qui jouera le rôle de la fonction objectif à minimiser. Le problème à optimiser devient :

$$\text{Minimiser } Y$$

sous les contraintes :

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

Cette transformation a une conséquence sur les performances de la résolution du problème à cause de l'ajout d'une variable et de N contraintes supplémentaires. La méthode utilisée généralement pour la résolution des problèmes linèaire en nombres entiers est la « séparation-évaluation » (ou branch and bound en anglais). Elle consiste en une application du fameux dicton « diviser pour régner », c'est-à-dire que si on ne peut pas résoudre directement un problème, on le divise en sous-problèmes plus simples. On construit ainsi un arbre de problèmes à résoudre qui s'agrandit avec le nombre de variables et le nombre des contraintes. L'optimisation linéaire en nombres entiers est un problème NP-difficile, et si la modélisation de notre problème en utilisant une norme L_2 a été résolue en un temps raisonnable, des problèmes de performance sont à craindre et nécessitent des expérimentations supplémentaires.

6.5 Conclusion

La mise en oeuvre de techniques de résumé appliquées à des courbes a permis de montrer que l'affectation de niveaux de résumé par optimisation linéaire permet de réduire significativement les erreurs de reconstruction par rapport à un résumé à niveau 'fixe'. L'approche proposée permet de limiter la surcharge du serveur central qui charge les données dans un entrepôt de donnée (datawarehouse). L'optimisation de l'affectation des niveaux de résumé est dynamique et s'adapte continuellement au contenu des flux de données. Ainsi, la technique de compression adoptée par chaque capteur dépend des variations des données dans le flux et est choisie parmi plusieurs approches. Notre problématique s'insère dans le cadre de résumé adaptatif de flux de données en spécifiant des fenêtres temporelles sautantes (courbe d'une période t).

Les expérimentations ont été réalisées sur des données réelles décrivant les mesures de consommation électrique générées par les compteurs d'un ensemble de clients. Toutes ces expérimentations montrent l'efficacité de notre outil générique de résumé optimisé de flux de données distribués. A l'issue de l'échantillonnage temporel, le serveur central récupère des courbes résumées qu'il faut reconstruire afin d'effectuer des opérations telles que l'agrégation au niveau central. Nous avons utilisé des méthodes très simples pour l'interpolation. L'objet du chapitre 8 est de présenter des techniques d'interpolation en prenant en compte des informations supplémentaires sur la courbe à reconstruire.

Chapitre 7

Echantillonnage spatio-temporel de flux de données distribués

Sommaire

7.1 Introduction

Nous rappelons le problème posé dans le cadre de cette thèse comme suit :

Nous disposons de N capteurs reliés à un serveur central et qui envoient une séquence de mesures temporelles en continu. Nous supposons que les données sont générées à des instants réguliers et que les mesures sont numériques et unidimensionnelles. Chaque flux est découpé en périodes (fenêtres temporelles) de même taille. Une fenêtre temporelle est constituée de p éléments (on suppose constant le nombre d'éléments d'une fenêtre car le flux est régulier). Nous utilisons le terme « courbe » pour qualifier chaque séquence temporelle. Nous supposons que le système central a une limite de stockage par période de temps à ne pas dépasser, soit s cette limite ($s < N * p$). L'objectif est d'approcher les courbes originales à partir d'un ensemble de données sélectionnées en respectant la limite de stockage au niveau central. Nous avons vu au chapitre 4 que plusieurs travaux ont été menés ces dernières années sur l'échantillonnage et la construction de résumés à partir de flux de données distribués. Deux approches peuvent naturellement être utilisées pour réduire la volumétrie des données courbes :

- L'échantillonnage temporel : conserver une partie des mesures sur tous les capteurs ;
- L'échantillonnage spatial : conserver toutes les mesures mais seulement d'une partie des capteurs.

Nous proposons dans ce chapitre une stratégie qui consiste à combiner ces deux approches, dans l'objectif des estimateurs des courbes agrégées de qualité acceptable (avec des erreurs d'estimation faibles).

7.2 Estimation des courbes échantillonnées

Nous expliquons dans cette section nos méthodes d'estimation des courbes à partir de l'échantillon construit soit temporellement soit spatialement.

7.2.1 Estimation de courbes échantillonnées temporellement

L'échantillonnage temporel consiste à conserver habilement un nombre réduit de mesures sur une courbe, sans perdre trop d'information sur la forme de la courbe. Si aucun échantillonnage temporel n'est réalisé, l'agrégation des courbes de la population consiste, à chaque instant t, à sommer l'ensemble des valeurs de consommation $C_i(t)$ mesurées sur chaque courbe i :

$$C(t) = \sum_{i=1}^{N} C_i(t) \tag{7.1}$$

L'échantillonnage temporel conduit à des courbes asynchrones (des mesures sont manquantes à certains instants) à partir desquelles nous allons effectuer des estimations (moyenne ou somme). La figure 7.1(a) montre trois séries de mesures représentées par les points triangle, carré et losange et qui sont très rarement mesurés aux mêmes instants. Ceci rend les calculs directs de moyennes ou de totaux non triviaux. Le fait d'échantillonner de manière temporelle les courbes nécessite donc de les reconstruire par interpolation à partir des mesures sélectionnées. Différentes approches d'interpolation peuvent être appliquées : linéaire, splines, ondelettes, etc [54; 101]. Ainsi, par exemple, une interpolation sur les 3 séries de mesures précédentes donnerait le résultat suivant (cf. figure 7.1(b))

Dans notre cas, nous proposons de remplacer dans l'équation 7.1 la valeur inconnue en un instant t par sa valeur interpolée :

$$\tilde{C}(t) = \sum_{i=1}^{N} \tilde{C}_i(t) = \sum_{i=1}^{N} (C_i(t) + \epsilon_i(t)) \tag{7.2}$$

Avec $\tilde{C}_i(t)$ la valeur interpolée à l'instant t et $\epsilon_i(t)$ l'erreur d'interpolation pour la courbe i à l'instant t.

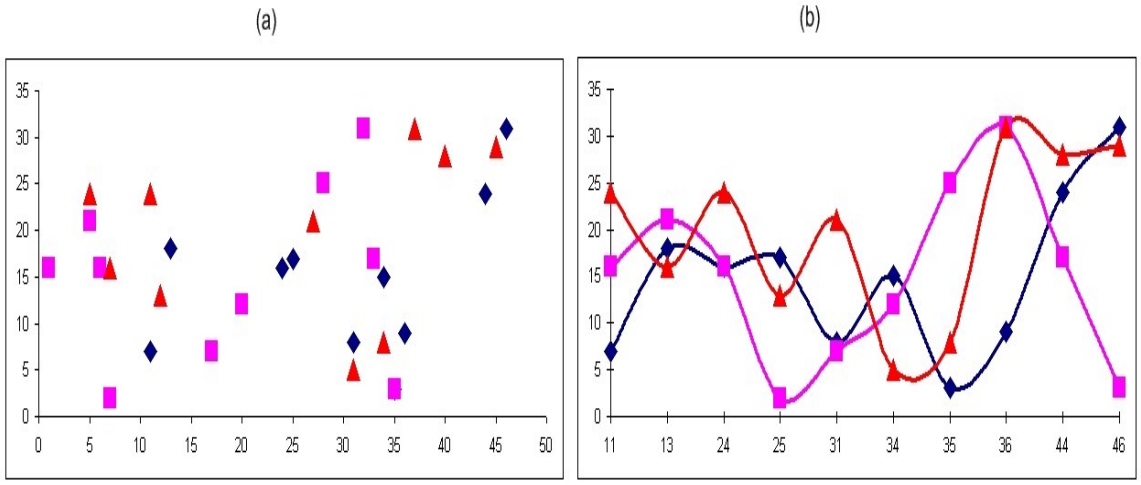

FIGURE 7.1 – mesures asynchrones et leurs courbes lissées

7.2.2 Estimation de courbes échantillonnées spatialement

L'échantillonnage spatial consiste à conserver un ensemble représentatif de courbes comme indiqué dans la figure 7.2 - ce qui revient à une problématique similaire à celles abordées dans la théorie des sondages.

Nous utilisons dans le cadre de ce travail un plan de sondage simple sans remise afin de tirer un échantillon aléatoire et uniforme de capteurs. Le seuil imposé s détermine la taille de l'échantillon, celle-ci se trouve donc limitée par la bande passante disponible, et sa valeur est de : $n = \left\lfloor \frac{s}{p} \right\rfloor$. Chaque capteur i a la même probabilité d'inclusion $\pi_i = \frac{n}{N}$. Et chaque échantillon S qui peut être formé a la même probabilité de sortie $p(S) = \binom{N}{n}^{-1}$.

Nous utilisons l'estimateur d'Horvitz Thompson qui estime sans biais la courbe globale :

$$\hat{C}_{HT}(t) = \sum_{i \in S} \frac{C_i(t)}{\pi_i} \tag{7.3}$$

Les courbes des capteurs ne faisant pas partie de l'échantillon sont donc estimées par la courbe moyenne de l'échantillon.

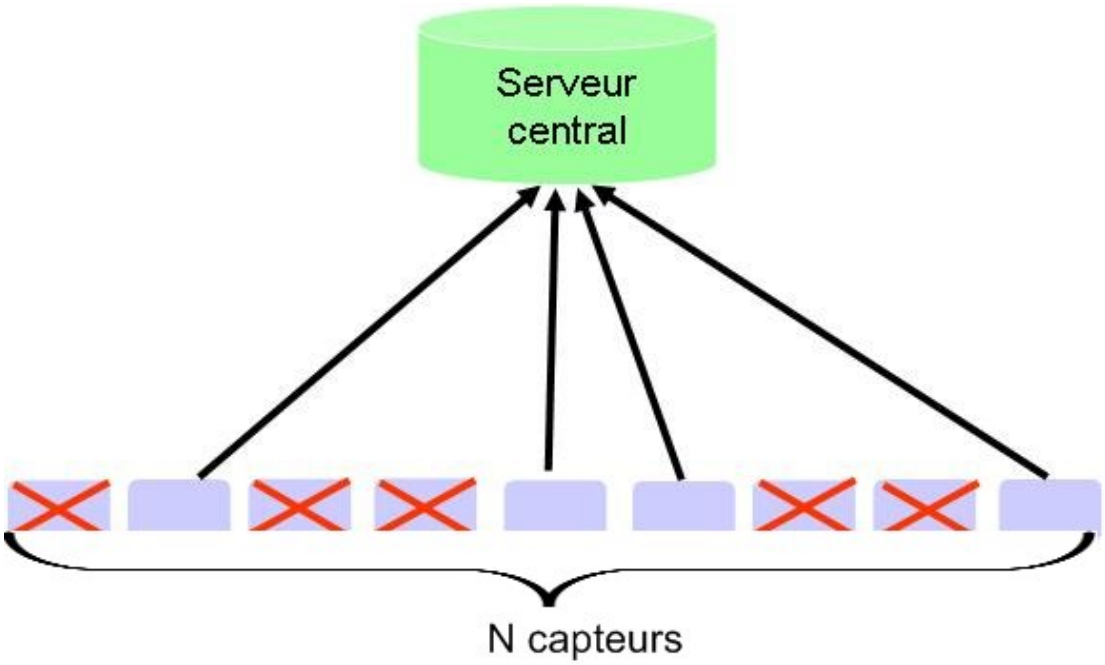

FIGURE 7.2 – Echantillonnage spatial : sélection aléatoire de n capteurs

7.3 Approche proposée et conséquences sur les estimations

7.3.1 Echantillonnage spatio-temporel

Dans les deux cas d'échantillonnage cités ci-dessus (échantillonnage temporel et échantillonnage spatial), nous n'utilisons pas les vraies valeurs résumées mais des estimations (interpolation dans le premier cas, estimations entachées d'une erreur d'échantillonnage dans le deuxième). Nous proposons une stratégie qui consiste à combiner ces deux approches. Nous nommerons cette stratégie échantillonnage spatio-temporel.

Nous utilisons notre approche d'échantillonnage temporel qui a été explicité dans le chapitre 6. Nous récupérons donc des courbes avec des granularités temporelles différentes, mais constantes tout au long de certaines périodes de chaque courbe. Pour plus de détail sur la modélisation et la résolution du problème, le lecteur peut se référer au chapitre 6. Nous appliquons ici un échantillonnage régulier comme méthode de résumé. Nous pouvons utiliser d'autres méthodes de résumé de courbes ou de séries temporelles telle que la segmentation, ou des résumés à base de décompositions fonctionnelles (sur des bases d'ondelettes par exemple). Nous reconstruisons les courbes échantillonnées par interpolation linéaire.

Nous proposons d'intégrer une approche sondage à notre programme d'échantillonnage temporel afin de constituer un échantillon spatio-temporel. Comme pour les approches traditionnelles, on établit un plan de sondage pour constituer un échantillon avec un nombre de capteurs n. Deux cas de figure peuvent se présenter :

- $n * p < s$ (figure 7.3 (a)) : la taille de l'échantillon n ne dépasse pas le seuil s et on est en mode dit *non saturé*. Les courbes de l'échantillon sont prises entièrement et on applique l'échantillonnage temporel au reste des capteurs (N-n) ne faisant pas partie de l'échantillon, avec un seuil de $s' = s - n * p$. Les courbes faisant partie de l'échantillon n'ont pas besoin d'être interpolées puisque nous récupérons la totalité des mesures. Par contre, il faut reconstruire chaque courbe ne faisant pas partie de l'échantillon par interpolation linéaire ou toute autre méthode. Nous utilisons pour l'interpolation la procédure suivante : nous estimons les données entre deux instants échantillonnés temporellement par la courbe moyenne de l'échantillon ajustée sur les points de ces deux instants.
- $n*p \geq s$ (figure 7.3 (b)) : la taille de l'échantillon n implique le dépassement du seuil s et on est en mode dit *saturé*. Dans ce cas on applique un échantillonnage temporel optimisé afin de récupérer des courbes à granularités variables dans le temps pour chaque capteur de l'échantillon. Les capteurs ne faisant pas partie de l'échantillon ne sont pas observés. Les courbes de l'échantillon sont reconstruites par interpolation linéaire. les courbes ne faisant pas partie de l'échantillon sont estimées comme dans le cas purement spatial, par la courbe moyenne de l'échantillon.

7.3.2 Conséquences sur les estimations

D'un point de vue formel, il suffit d'estimer une courbe globale à partir d'un échantillon S. Nous pouvons élaborer un estimateur à partir d'un estimateur de type Horvitz-Thompson :

$$\hat{\tilde{C}}_{HT}(t) = \sum_{i \in S} \frac{\tilde{C}_i(t) + \epsilon_i(t)}{\pi_i} \tag{7.4}$$

La principale question qui se pose à nous à ce stade est : la stratégie « spatio-temporelle » proposée précédemment peut-elle être plus performante qu'une stratégie avec échantillon simplement spatial ou simplement temporel ? D'une manière évidente, la réponse va dépendre de la stratégie d'échantillonnage spatial choisie, mais elle peut aussi dépendre des techniques de redressement habituellement utilisées en sondage pour améliorer l'estimateur (au sens de la réduction de la variance d'échantillonnage). Elle va dépendre aussi de la technique d'interpolation utilisée.

Le fait de ne pas travailler avec les vraies mesures, mais avec des mesures entachées d'une erreur aléatoire, que nous supposerons maîtrisée, met à mal le cadre général d'application de l'estimateur d'Horvitz Thompson, en espérance et en variance. Le caractère non biaisé, et la formule d'estimation de sa variance présentée à la section 4.2.3 ne sont valables que si les mesures effectuées sur l'échantillon sont exactes, ce qui n'est pas le cas ici de par la présence de $\epsilon_i(t)$. Il nous faut donc connaître les

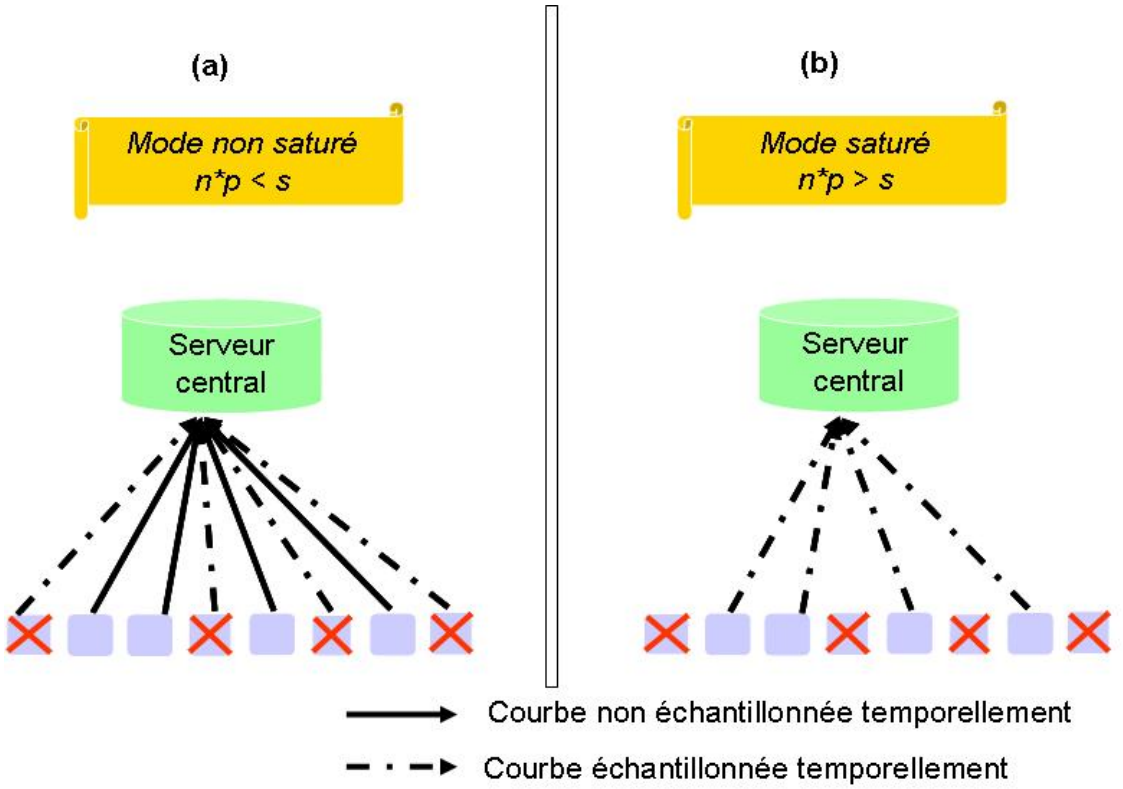

FIGURE 7.3 – Echantillonnage spatio-temporel

performances de l'estimateur 7.4. En effet, quelques propositions concernant cet estimateur ont été prouvées par Alain Dessertaine [53] sous l'hypothèse d'indépendance entre les erreurs d'interpolation des courbes et le plan de sondage. Cette hypothèse est traduite par

$$\forall(i,t) \quad Cov(\epsilon_i(t), \mathbf{1}_{i\in S}) = 0) \tag{7.5}$$

avec $\mathbf{1}_{i\in S}$ la variable indicatrice d'appartenance à l'échantillon S, d'espérance π_i.

Proposition 7.3.2.1 *L'estimateur Horvitz-Thompson 7.4 estime sans biais la courbe globale en chaque instant t.*

Proposition 7.3.2.2 *La variance de l'estimateur Horvitz-Thompson sur les valeurs interpolées devient :*

$$\hat{Var}(\hat{\tilde{C}}_{HT}(t)) = \sum_{i\in S}\sum_{j\in S}\left(\frac{\tilde{C}_i(t)}{\pi_i}\frac{\tilde{C}_j(t)}{\pi_j}(\pi_{ij}-\pi_i\pi_j)\right) + \sum_{i\in S}\frac{\sigma_i^2(t)}{\pi_i} \tag{7.6}$$

Avec $\sigma_i(t)$ la variance du résidu $\epsilon_i(t)$ en chaque instant, que nous supposerons connue.

Pour les démonstrations de ces deux propositions, le lecteur peut se référer à l'article de Dessertaine [53].

Le premier élément de la somme de la formule (7.6) peut être particulièrement bien maîtrisé, et la deuxième partie de ce calcul pourra être d'autant plus faible que nous pourrons, sur les courbes échantillonnées, utiliser un nombre de points de mesures plus important que dans le cas d'un recensement. Mais une difficulté supplémentaire apparaît à ce niveau. En effet, l'optimisation de l'échantillonnage temporel doit se faire sur l'échantillon spatial choisi. Aussi, l'erreur d'interpolation en un instant t d'une courbe de l'échantillon dépend non seulement de la technique d'interpolation utilisée, mais aussi de l'optimisation temporelle, et donc de l'échantillon sur lequel l'optimisation est utilisée. Ainsi, nous voyons apparaître dans l'estimateur Horvitz-Thompson deux niveaux d'aléas :

1. Le plan de sondage
2. Le caractère aléatoire de l'interpolation des courbes conditionnellement à l'échantillon (et conditionnellement à la technique d'interpolation utilisée).

Pour ce faire, nous pouvons envisager de combiner l'utilisation d'un boostrap adapté pour prendre en compte l'effet du plan de sondage utilisé, et des trajectoires probables de nos courbes lissées entre deux mesures en imposant certaines contraintes naturelles à imposer à nos trajectoires en utilisant des générateurs Browniens comme dans Abdessalem et al. [4].

7.4 Expérimentations

Nous avons expérimenté les approches décrites dans le paragraphe précédent grâce au second jeu de données (chapitre 5) composé de 1000 courbes relevées à intervalle de 10 minutes pendant une journée (144 mesures par courbe). Nous avons expérimenté plusieurs valeurs de seuil s (nombre total de données que le serveur central peut accepter pendant une période donnée). Pour calculer les erreurs commises dans le cas de l'échantillonnage spatial de capteurs, nous avons effectué des simulations de Monte-Carlo (100 simulations).

Les résultats sont donnés sous forme de courbes dans la figure 7.4. Nous avons calculé la moyenne (ST), le maximum (STmax) et le minimum (STmin) des erreurs quadratiques globales commises lors des simulations dans le cas de l'échantillonnage Spatio-Temporel, et nous avons calculé les erreurs quadratiques globales (T) dans le cas de l'échantillonnage purement Temporel. L'abscisse dans les figures correspond au taux de compression appliqué à l'ensemble des points des courbes de consommation : un taux de compression de 2 signifie que nous récupérons $s = \frac{(144*1000)}{2}$ points. Nous avons fait varier la taille de l'échantillon n dans le cas de l'échantillonnage spatio-temporel.

Les courbes à gauche de la ligne verticale sur la figure correspondant à une taille de l'échantillon $n = 200$ indiquent que nous sommes en mode *non saturé* pour l'échantillonnage spatio-temporel. A droite de la ligne, la taille n implique un dépassement du seuil ainsi que pour la figure de droite correspondant à une taille de l'échantillon de $n = 800$ (mode *saturé*). Nous remarquons que l'échantillonnage spatio-temporel permet de réduire en moyenne les erreurs commises par rapport à un échantillonnage purement temporel. Ce dernier s'avère meilleur dans le cas d'un échantillon de

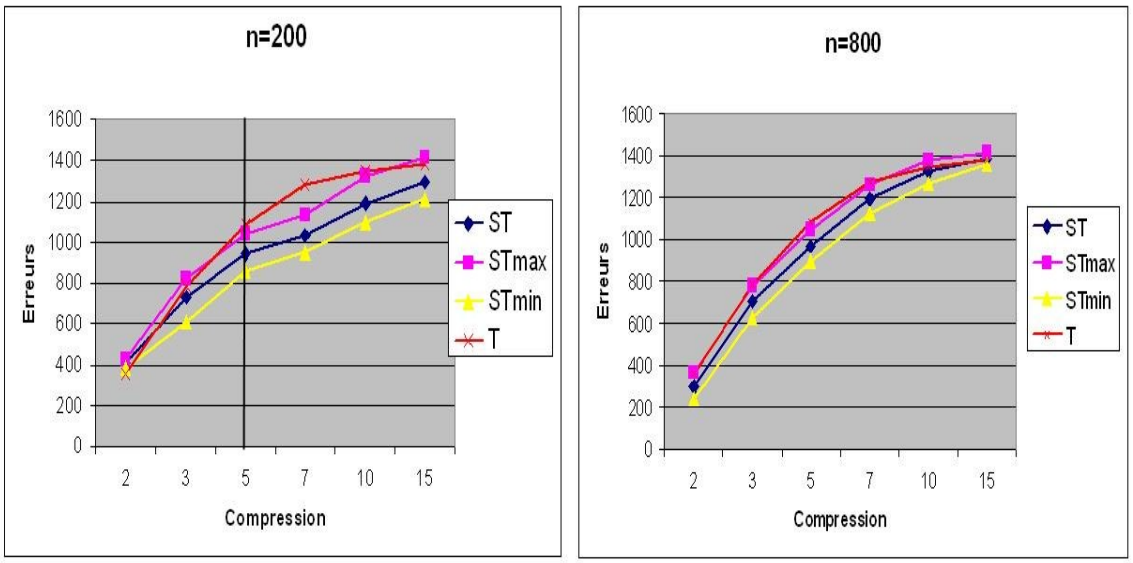

FIGURE 7.4 – Echantillonnage spatio-temporel Vs. Echantillonnage purement temporel

petite taille et un faible taux de compression. Toutefois, les deux courbes (STmax) correspondant au maximum des erreurs et (T) correspondant aux erreurs par échantillonnage purement temporel, se confondent et ceci pour toute valeur de n et tout taux de compression. A noter que la méthode de tirage effectuée est un échantillonnage aléatoire uniforme. L'utilisation d'autres méthodes telle que l'échantillonnage stratifié, avec comme variables de stratification des données commerciales (équipements des clients, tarification,etc.), données géographiques, ou encore le niveau de consommation électrique aurait pu améliorer les résultats par rapport à la méthode simple.

La figure 7.5 montre les erreurs correspondant à l'échantillonnage purement spatial (S), l'échantillonnage purement temporel (T) et l'échantillonnage spatio-temporel (ST) avec une taille d'échantillon de $n = 200$. Nous avons expérimenté plusieurs valeurs de taux de compression. La taille de l'échantillon spatial dépend de ce dernier. Un taux de compression de 2 signifie que nous tirons un échantillon aléatoire uniforme de capteurs de taille $n = \frac{1000}{2}$. Les courbes (S) et (ST) correspondent aux courbes moyennes des erreurs quadratiques commises lors des simulations. La figure 7.5 montre clairement que l'échantillonnage spatial permet d'estimer mieux la courbe de consommation globale que l'échantillonnage purement temporel et celà pour presque tous les taux de compression. Nous remarquons aussi que l'échantillonnage purement spatial a un comportement plus ou moins similaire à l'échantillonnage spatio-temporel avec une légère amélioration pour les faibles taux de compression. Les erreurs d'interpolation sont donc plus importantes que les erreurs d'échantillonnage dans le cas de nos courbes d'expérimentation.

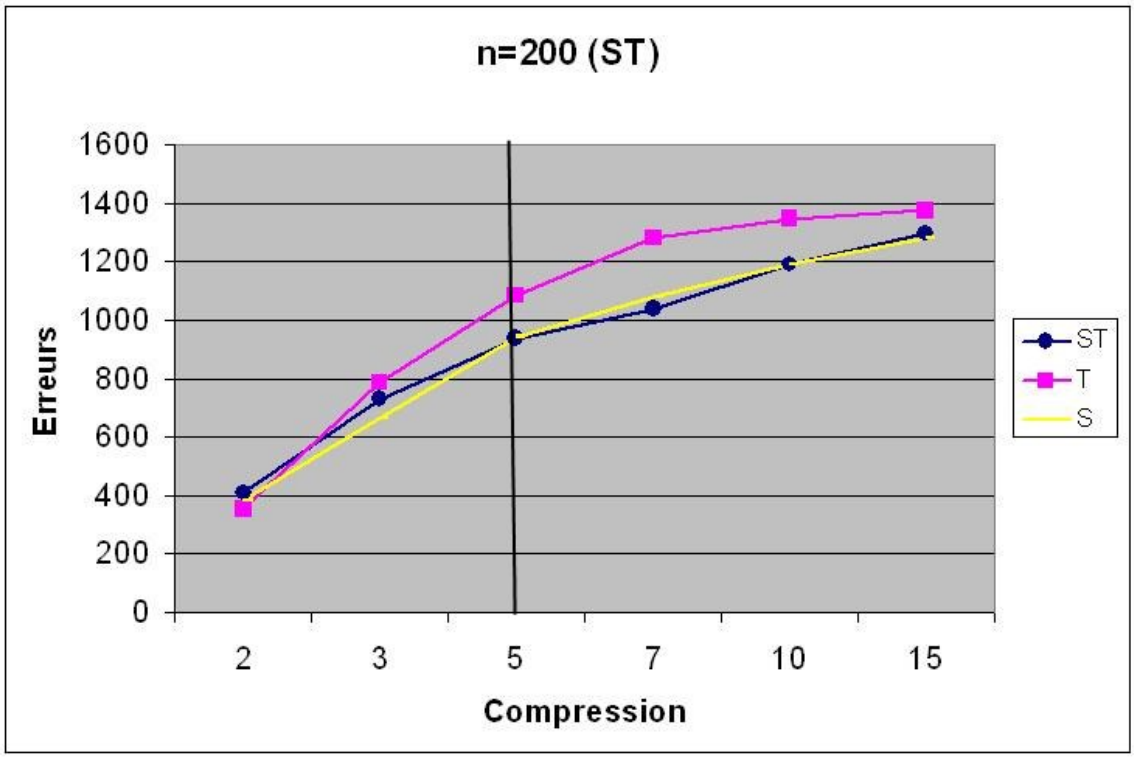

FIGURE 7.5 – Echantillonnage purement spatial Vs. Echantillonnage purement temporel Vs. Echantillonnage spatio-temporel de taille n=200

7.5 Conclusion

Nous avons proposé une approche d'échantillonnage spatial de capteurs distribués couplée à un échantillonnage temporel. Nos expérimentations ont montré l'intérêt de cette approche par rapport à un échantillonnage temporel exhaustif (appliqué à tous les capteurs). Nous avons aussi montré que l'échantillonnage spatial s'avère meilleur pour estimer une courbe globale. Ces expérimentations ont été faites de façon empirique. Dans le prochain chapitre 8, nous proposerons des approches permettant d'estimer les erreurs d'interpolation à chaque instant. Ceci nous permettra de comparer ces trois méthodes dans le cas d'estimation des courbes individuellement ou sur des petits domaines et de comparer leur comportement dans ce contexte.

Ce travail peut être inséré dans une approche panel des flux de données provenant de capteurs. Ceci permet de rendre dynamique le panel en pilotant la fréquence d'observation des éléments des flux en fonction des contraintes de l'infrastructure et des besoins en terme d'analyse. Un panel dynamique est d'autant plus intéressant par rapport à un contexte de panel standard du fait de la richesse des données prélevées :

– L'information que nous cherchons est directement prélevée, en même temps que l'échantillon se construit. Elle peut donc être utilisée, d'une manière adaptative, pour améliorer le plan d'échantillonnage. Cette amélioration peut s'effectuer

soit directement, soit indirectement en imaginant l'échantillon des flux comme une première phase à l'échantillon final.
- Le caractère 'continu' des flux et l'aspect temporel des données pourra, entre autre, capter au fil de l'eau les variations mêmes de la population d'étude.

Chapitre 8

Interpolation de courbe connaissant son intégrale

Sommaire

8.1 Introduction

Les travaux présentés dans ce mémoire de thèse peuvent être appliqués aux flux de données provenant des compteurs électriques afin de les échantillonner temporellement, spatialement ou bien spatio-temporellement. L'approche temporelle et spatio-temporelle de résumé de flux de données distribués conduit à des courbes asynchrones. Ce qui signifie qu'à un instant donné, des mesures seront manquantes au niveau de quelques capteurs. Il est donc nécessaire de reconstruire par interpolation les mesures manquantes afin d'agréger les flux de données. L'interpolation induit des erreurs non négligeables. Toutes les expérimentations qui ont été décrites jusque là ont été réalisées de façon empirique. De plus, nous avons supposé maîtriser

les erreurs d'interpolation, notamment dans l'expression de la variance d'Horvitz-Thompson qui est :

$$\hat{Var}(\hat{\tilde{C}}_{HT}(t)) = \sum_{i \in S} \sum_{j \in S} \left(\frac{\tilde{C}_i(t)}{\pi_i} \frac{\tilde{C}_j(t)}{\pi_j} (\pi_{ij} - \pi_i \pi_j) \right) + \sum_{i \in S} \frac{\sigma_i^2(t)}{\pi_i}$$

Dans ce chapitre, nous proposons des méthodes d'estimation à tout instant t de la courbe de charge échantillonnée ainsi que sa variance $\sigma(t)$ à l'instant t en prenant en compte une propriété supplémentaire connue : son intégrale.

8.2 Motivations

Afin de répondre aux requêtes d'agrégation de l'ensemble ou d'un sous-ensemble de courbes de consommation électrique par l'approche spatiale, on peut utiliser l'estimateur d'Horvitz-Thompson. Nous rappelons ici son expression :

$$\hat{C}_{HT}(t) = \sum_{k \in S} \frac{C_k(t)}{\pi_k} \tag{8.1}$$

Où $\pi_k = \frac{n}{N}$, n est la taille de l'échantillon S et N la taille de la population.

Sa variance peut être estimée par

$$\hat{Var}(\hat{C}_{HT}(t)) = \frac{N(N-n)}{n(n-1)} \sum_{j \in S} (C_j(t) - \frac{1}{n} \sum_{k \in S} C_k(t)) \tag{8.2}$$

L'intervalle de confiance peut être estimé pour un risque α par l'expression suivante :

$$CI_{(1-\alpha)}(t) = [\hat{C}_{HT}(t) - z_{1-\alpha/2} \sqrt{\hat{Var}(\hat{C}_{HT}(t))} \\ , \hat{C}_{HT}(t) + z_{1-\alpha/2} \sqrt{\hat{Var}(\hat{C}_{HT}(t))}] \tag{8.3}$$

Où $z_{1-\alpha/2}$ est le quantile $(1-\alpha/2)$ d'une variable aléatoire de distribution normale.

Quant à l'approche temporelle, le problème posé est tout d'abord de reconstituer (ou d'approcher) une série temporelle à partir de la connaissance de ses valeurs en un nombre limité de points. L'interpolation [97] et la régression [85; 113] permettent d'apporter une solution à ce problème. La différence entre ces deux techniques est que l'interpolation désigne une fonction qui passe exactement par les points qu'on connaît, alors que la régression est une fonction qui s'approche le plus possible selon un critère donné des points sans forcément passer par eux. On utilise généralement le critère des moindres carrés. La régression se justifie quand en pratique, les observations sont bruitées, ceci peut provenir d'une imprécision de mesure par exemple.

La qualité d'un procédé d'interpolation (ou de régression) se mesure par une estimation d'erreur. Il s'agit de vérifier si la fonction interpolante (ou le modèle de régression) utilisée approche bien la série qu'on cherche à reconstruire. On trouve dans

la littérature le terme résidu pour qualifier l'erreur d'interpolation (ou de régression). Ces méthodes d'approximation dépendent d'un certain nombre d'hypothèses sur le résidu qui ne sont pas généralement vérifiées. Parmi ces hypothèses : le résidu est une variable aléatoire de moyenne nulle et de variance constante (c'est ce qu'on appelle homogénéité de la variance ou *homoscédasticité*). La plupart des résultats reposent explicitement ou implicitement sur ces deux hypothèses d'homoscédasticité et de normalité. Or, dans la pratique ceci n'est pas toujours vrai. Récemment, d'importants approfondissements ont vu le jour dans la littérature pour modéliser le phénomène où les résidus varient dans le temps, c'est ce qu'on appelle *hétéroscédasticité* [30; 31]. Toutefois, à notre connaissance, aucune des méthodes proposées dans la littérature ne prennent en compte à la fois la série temporelle (la courbe) à estimer et son intégrale comme il est le cas dans notre problématique. En effet, nous travaillons sur des courbes de charge, ce sont des courbes représentant l'évolution de la consommation électrique en puissance d'un client. Chaque courbe de charge est composée de mesures relevées par intervalle de temps qui peut être une seconde, une minute, 10 minutes, etc. et sa longueur peut être d'une journée, un mois, un an, etc.

Nous rappelons que le schéma du flux de consommation électrique est sous la forme (t,$C(t)$,$E(t)$) avec t le timestamp, $C(t)$ la mesure en puissance et $E(t)$ est la mesure d'énergie consommée jusqu'à l'instant t (voir chapitre 5).

Nous considérons une courbe C pour laquelle les mesures aux instants t_a et t_b ont été relevées par le serveur central, mais celles situées entre t_a et t_b sont manquantes à cause de l'échantillonnage temporel. Nous cherchons donc à les déterminer en considérant les caractéristiques de la courbe citées ci-dessous :

1. Les puissances de la courbe aux deux instants $C(t_a)$ et $C(t_b)$ sont connues, car elles font parties de l'échantillon temporel
2. Les puissances pour tout instant entre t_a et t_b sont des nombres réels positifs

$$\forall t \in]t_a, t_b[\quad C(t) \geq 0$$

3. Les puissances entre t_a et t_b ne doivent pas dépasser une valeur maximale

$$\exists c_{max} \quad \forall t \in]t_a, t_b[\quad C(t) \leq c_{max}$$

4. L'énergie E_{ab} consommée entre t_a et t_b est connue et correspond à l'intégrale sous la courbe entre t_a et t_b :

$$\int_{t_a}^{t_b} C(t)dt = E_{ab}$$

Nous cherchons à prendre en compte ces propriétés afin de proposer une interpolation de la courbe échantillonnée et d'estimer l'intervalle de confiance autour de cette interpolation. Nous détaillons dans ce qui suit deux approches l'une dite *naïve* et l'autre dite *stochastique* permettant de résoudre ce problème. Notons que nous travaillons avec des intervalles de temps discrets, ainsi les instants entre t_a et t_b peuvent être représentés par l'ensemble :$\{t_a + 1, t_a + 2, ..., t_b - 1\}$

8.3 Approche naïve

Nous cherchons à estimer les points situés entre t_a et t_b par interpolation, en tenant compte des propriétés de la courbe décrites dans la section 8.2. Nous proposons une approche basée sur les données historiques de la consommation : chaque capteur calcule la distribution des pentes entre deux mesures de charges successives à partir de ses données du passé. Une pente entre deux points à deux instants consécutifs $t-1$ et t est définie par

$$C(t) - C(t-1)$$

A la fin de chaque période, le capteur envoie la *pente inférieure* notée α_{min} et la *pente supérieure* notée α_{max} correspondant à la probabilité qu'une pente aléatoire α soit dans l'intervalle spécifié par $[\alpha_{min}, \alpha_{max}]$ soit égale à une valeur X fixée par l'utilisateur, ce qui signifie :

$$P(\alpha_{min} \leq \alpha \leq \alpha_{min}) = X$$

Le serveur central utilise ces pentes dans des problèmes d'optimisation linéaire afin de calculer l'interpolation de la courbe résumée et d'estimer l'*enveloppe* correspondant à l'espace des solutions potentielles. Dans nos expérimentations, la distribution de α s'est avérée être une distribution normale. Les probabilités souvent utilisées dans ce cas sont :

- $X = 0.68$, signifie que 68% des observations (pentes) s'écartent de la moyenne σ d'un écart-type μ, c-à-d entre $\mu - \sigma$ et $\mu + \sigma$;
- $X = 0.95$, signifie que 95% des observations (pentes) s'écartent de la moyenne σ de deux écart-types μ, c-à-d entre $\mu - 2\sigma$ et $\mu + 2\sigma$;
- $X = 0.997$, signifie que 99.7% des observations (pentes) s'écartent de la moyenne σ de trois écart-types μ, c-à-d entre $\mu - 3\sigma$ et $\mu + 3\sigma$.

8.3.1 Estimation de l'erreur

Cette section décrit la méthode utilisée afin de construire l'enveloppe des courbes possibles entre t_a et t_b en respectant les contraintes de valeurs limitées de C et la valeur connue de l'intégrale. Cette enveloppe représente une estimation de l'erreur d'interpolation pour tous les instants entre t_a et t_b. L'idée est : connaissant α_{min} et α_{max}, la valeur $C(t_a + 1)$ ne peut pas être en dehors de l'intervalle $[C(t_a) + \alpha_{min}$, $C(t_a) + \alpha_{max}]$ et ainsi de suite jusqu'à arriver à t_b. Nous ajoutons la contrainte des bornes sur C et de l'intégrale connue, ce qui donne les deux programmes d'optimisation suivants qui correspondent aux enveloppes minimum et maximum :

$$\text{Pour chaque } t \in \{t_a + 1, t_a + 2, ..., t_b - 1\} \text{ Minimiser et Maximiser } C(t)$$

Sous les contraintes :

$$\begin{cases} C(t-1) + \alpha_{min} \leq C(t) \leq C(t-1) + \alpha_{max} \\ 0 \leq C(t) \leq c_{max},\ t \in \{t_a + 1, t_a + 2, ..., t_b - 1\} \\ \sum_{t_a}^{t_b} C(t) = E \\ t \in]t_a, t_b[\end{cases}$$

La première contrainte définit α_{min} (α_{max} respectivement) comme la pente minimale (maximale) entre deux points consécutifs que nous cherchons à estimer. La seconde contrainte impose des valeurs positives et une puissance maximale aux mesures de consommation. La troisième contrainte assure que les points interpolés respectent la contrainte d'énergie. Ces problèmes se résolvent facilement et dans un temps limité grâce aux techniques connues de la programmation linéaire tel que le simplexe [70].

8.3.2 Reconstruction de la courbe

Nous avons décrit dans la sous-section qui précède, la technique utilisée pour estimer l'enveloppe des solutions. Nous allons maintenant présenter notre approche d'interpolation d'une courbe de charge entre deux points échantillonnés. Le principe de notre approche est de réduire l'enveloppe au maximum jusqu'à obtenir une unique solution (une seule pente entre deux points à interpoler) qui respecte les contraintes et qui donnera la solution de l'interpolation.

Afin de trouver une courbe interpolée entre deux instants sélectionnés t_a et t_b qui respecte les caractéristiques de la courbe, on résout le problème suivant :

$$\text{Minimiser } \alpha$$

Sous les contraintes :

$$\begin{cases} C(t) - C(t-1) = \alpha, \quad t \in \{t_a + 1, t_a + 2, ..., t_b - 1\} \\ 0 \leq C(t) \leq c_{max}, \quad t \in \{t_a + 1, t_a + 2, ..., t_b - 1\} \\ \sum_{t_a}^{t_b} C(t) = E \\ \alpha_{min} \leq \alpha \leq \alpha_{max} \end{cases}$$

Il s'agit d'un problème d'optimisation linéaire visant à minimiser une pente en respectant les contraintes sur la courbe de charge. L'objectif de ce programme est de faire converger la pente minimale α_{min} et la pente maximale α_{max} vers une unique pente α qui est solution du problème d'optimisation linéaire. La première contrainte définit α comme une pente entre deux points à interpoler. La seconde contrainte signifie que les mesures de consommation sont positives et ne dépassent pas une puissance maximale. La troisième contrainte correspond à la contrainte d'énergie. Et enfin, la dernière contrainte limite la recherche de α dans l'intervalle $[\alpha_{min}, \alpha_{max}]$.

Il peut arriver qu'aucune solution ne satisfasse les programmes d'optimisation ci-dessus. Ceci peut se produire si la courbe change subitement de tendance par rapport à son passé. Dans ce cas, nous considérons que l'enveloppe des solutions et nulle et nous résolvons le problème d'optimisation linéaire précédent sans la dernière contrainte, afin de trouver la pente minimale qui permet de donner une reconstruction de la courbe sous la contrainte de l'énergie et de la puissance maximale.

8.3.3 Exemple

Nous utilisons une courbe de charge constituée de 48 mesures (un point de mesure à chaque demi-heure) comme exemple de notre approche. La courbe originale

correspondant aux 5 premières valeurs est donnée par :
$C = \{31.67, 30.33, 24.33, 23, 28\}$
Nous estimons α_{min} et α_{max} à partir des courbes de charge historiques des 100 jours précédents. Nous calculons les pentes entre deux points consécutifs et nous construisons la distribution de ces pentes, représentée dans la figure 8.1. Nous observons que les pentes suivent une loi quasi-normale, avec une moyenne approchant de 0 (pas de tendance dans la consommation de ce client). Les pentes $\alpha_{min} = -8.18$ et $\alpha_{max} = 8.18$ correspondent à la probabilité égale à 68% qu'une pente α soit comprise dans l'intervalle $[\alpha_{min}, \alpha_{max}]$.

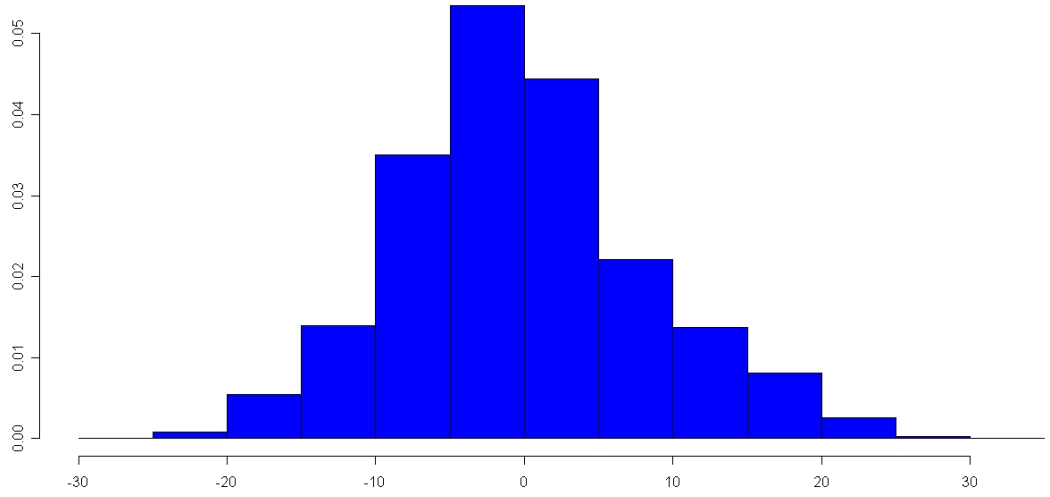

FIGURE 8.1 – Histogramme des pentes passées

Estimation entre deux valeurs sélectionnées

Supposons qu'un échantillonnage temporel a affecté un pas égal à 5 à la courbe C, ce qui signifie que nous gardons les points $C(0) = 31.67$ et $C(4) = 28$, et que nous devons estimer les points intermédiaires en prenant en compte l'énergie consommée pendant les trois heures $E = 109.33$. Supposons que la valeur maximale à ne pas dépasser est $C_{max} = 250kW$.

Nous résolvons les problèmes linéaires définis précédemment, il y en a 7 en tout, 2 pour chacun des 3 points intermédiaires à estimer afin de déterminer l'enveloppe, et le dernier problème sert à déterminer la pente pour l'interpolation. Les solutions à ces problèmes sont représentées dans la figure 8.2 : La courbe originale (LC) est représentée en continu, l'interpolation (Inter.) en tiret et l'enveloppe (Env.) en pointillé. Nous avons calculé le temps écoulé afin d'estimer l'enveloppe et la courbe sur une machine Intel Core2 CPU 1.20GHZ et 1Go de mémoire et nous avons obtenu

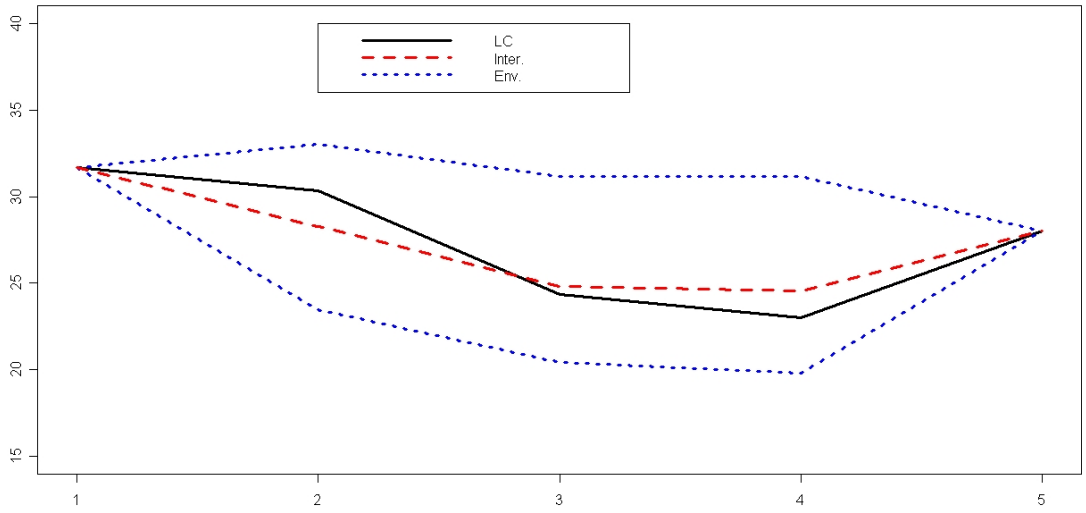

FIGURE 8.2 – Estimation de la courbe et de l'enveloppe(Estimation de l'erreur) entre deux points

en moyenne 0.28 secondes. Nous précisons que les programmes ont été implémentés avec le langage R.

Estimation de la courbe

Nous allons maintenant présenter les résultats de l'approche sur une courbe d'une journée, montrant l'interpolation réalisée avec deux taux d'échantillonnage différents. Dans ces deux expériences, des taux d'échantillonnage de 5 (Figure 8.3(a)) et 20 (Figure 8.3(b)) ont été utilisés.

La partie gauche de la figure 8.3 montre un zoom des courbes de droite afin de décrire plus précisément la courbe originale et son interpolation grâce à l'approche naïve. Les parties de droite de la figure montrent les enveloppes entre chacune des valeurs collectées. Il est clair que pour les deux taux d'échantillonnage, la courbe interpolée est très proche de l'original. Nous notons également que plus le taux de compression est élevé plus l'enveloppe est grande.

Toutefois, les taux d'échantillonnage sont affectés en utilisant une technique d'optimisation qui donne un taux d'échantillonnage plus grand aux courbes qui varient légèrement. L'enveloppe dans ce cas donne une estimation très pessimiste des erreurs. En effet, l'approche naïve ne fournit pas la probabilité que la courbe soit à l'intérieur de l'enveloppe. Une perspective de ce travail sera d'étudier la relation entre ces enveloppes et la variance de l'estimation à chaque point interpolé. Dans la prochaine section, nous allons décrire une seconde approche basée sur un modèle stochastique afin de pallier au problème de l'intervalle de confiance pour lequel l'approche naïve ne donne pas d'estimation exacte.

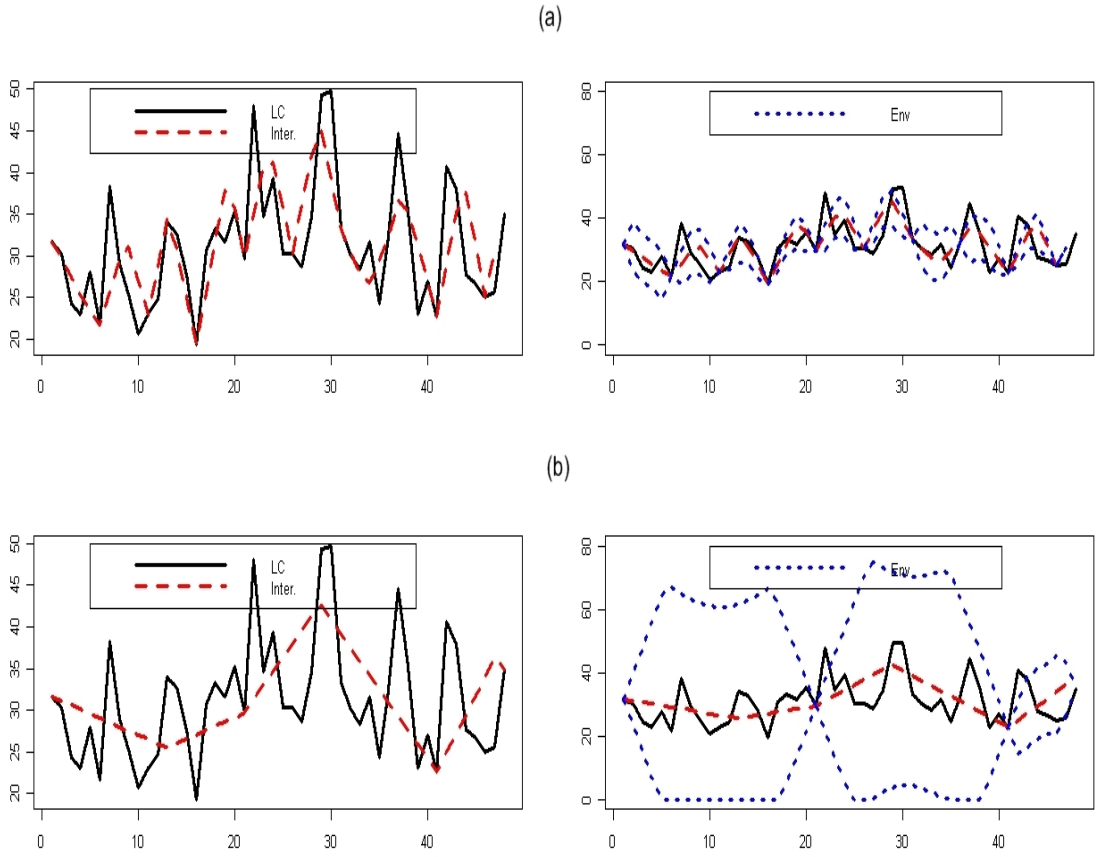

FIGURE 8.3 – Estimation de la courbe et de l'enveloppe(Estimation de l'erreur)

8.4 Approche stochastique

L'idée à l'origine de l'interpolation de la courbe de consommation électrique par un processus stochastique est que l'on peut modéliser les fluctuations de la consommation comme un phénomène aléatoire. En effet, une variation dans la courbe de charge est induite par le déclenchement d'une activité domestique tels que la préparation de repas, le lancement d'une machine à laver, l'extinction d'un radiateur électrique, etc. Les transitions entre ces activités peuvent être modélisées par un processus de Markov [57]. Nous supposons que l'effet global des fluctuations de la consommation peut être résumé par des fluctuations browniennes. De plus, comme une consommation électrique est toujours positive, ceci nous amène à considérer un processus positif avec un comportement brownien. Le plus simple de tous est ce qu'on appelle le mouvement brownien géométrique, qui est également le modèle de base pour l'évolution des actifs en mathématiques financières.

8.4.1 Mouvement brownien géométrique

Le but de cette section est de présenter les modèles de base servant comme « point de départ » à notre modèle d'interpolation. Essentiellement, nous nous intéressons à cinq types de processus célèbres allant du plus général au plus spécifique [33].

1. Processus de Markov : C'est un processus stochastique où la prédiction du futur à partir du présent ne nécessite pas la connaissance du passé. En effet, l'état du processus markovien au temps t ne dépend que de son état au temps $t-1$;
2. Processus brownien : (ou processus de Wiener) B_t est un processus de Markov particulier évoluant de façon continue. Le processus brownien est nommé ainsi en hommage au botaniste Robert Brown, qui a décrit le mouvement d'une particule soumise à une infinité de chocs en des temps très courts. On dit que B_t suit un processus brownien si $dB_t = \epsilon * \sqrt{(dt)}$ où ϵ une variable aléatoire normale suivant la loi $N(0,1)$. Les valeurs de dB_t pour 2 périodes différentes sont indépendantes les unes des autres. La figure 8.4 est un exemple d'un processus brownien. Ce dernier a les propriétés suivantes :
   - $B_0 = 0$ et B_t évolue de façon continue
   - Le processus B_t est à accroissements indépendants
   - $(B_{t+\delta t} - B_t)$ suit une loi normale centrée de variance δt
     - $E[B_T - B_0] = 0$
     - $Var[B_T - B_0] = T$

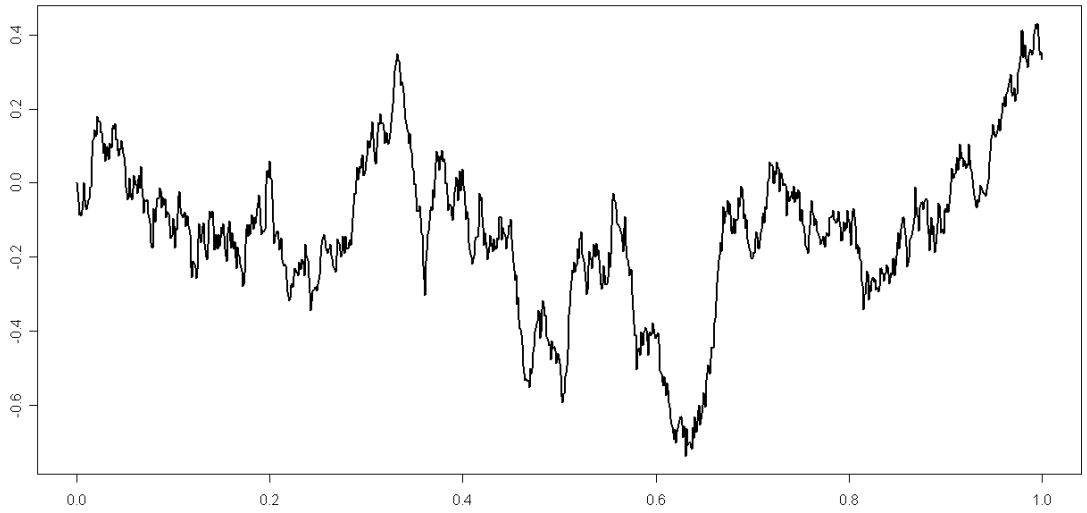

FIGURE 8.4 – Exemple d'un processus brownien

3. Processus brownien généralisé : Dans un processus brownien généralisé X_t, on peut fixer librement un taux de dérive (drift) pour le processus ainsi qu'une

variance. Formellement, on l'écrit sous la forme :

$$\underbrace{dX_t = a.dt}_{(1)} + \underbrace{b.dB_t}_{(2)}$$

Avec
(1) 'a' est le drift, c'est la variation moyenne par unité de temps du processus.
(2) dB_t est l'accroissement d'un processus brownien standard. On ajoute un bruit stochastique à la trajectoire, c'est cette partie de l'équation qui crée la variabilité du processus.
La figure 8.5 montre un exemple d'un mouvement brownien généralisé avec $a = 3$, $X_0 = 5$ et $b = 2$. Les propriétés d'un mouvement brownien généralisé sont :
– Espérance : $E[X_T - X_0] = aT$
– Variance : $Var[X_T - X_0] = b^2T$

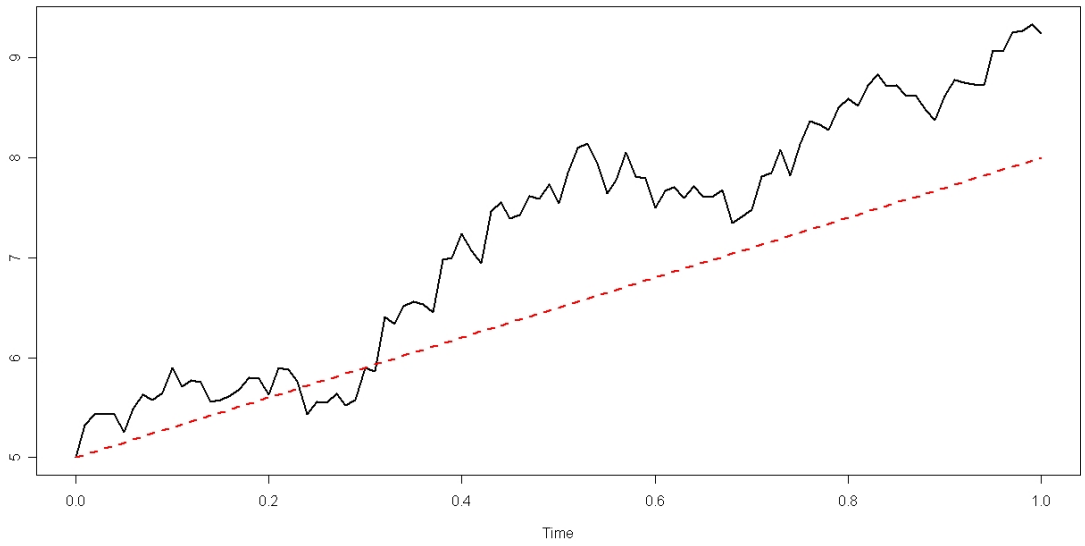

FIGURE 8.5 – Exemple d'un mouvement brownien généralisé avec $a = 3$, $X_0 = 5$ et $b = 2$

4. Processus d'Itô : est un processus brownien généralisé pour lequel 'a' et 'b', c'est-à-dire le drift et la quantité de bruit dépendent eux-mêmes du temps

$$dX_t = a(X_t, t)dt + b(X_t, t)dB_t$$

5. Mouvement brownien géométrique : est un cas particulier du processus d'Itô X_t. Il est souvent utilisé en finance comme le plus simple modèle d'évolution de cours de bourse. Il s'agit de la solution de l'équation différentielle stochastique :

$$dX_t = \alpha X_t dt + \beta X_t dB_t$$

Avec $a(X_t, t) = \alpha X_t$ et $b(X_t, t) = \beta X_t$. Grâce au lemme d'Itô, on peut calculer la solution de l'équation différentielle du mouvement brownien géométrique. Le but du lemme d'Itô est de déterminer le processus suivi par toute fonction $G(x, t)$ du temps t et d'une variable x qui suit un processus d'Itô (dont le processus brownien géométrique).

Lemme d'Itô : $G(X, t)$ est caractérisé par le processus suivant

$$dG = (\frac{\partial G}{\partial X}a + \frac{\partial G}{\partial t} + \frac{1}{2}\frac{\partial^2 G}{\partial^2 X}b^2)dt + \frac{\partial G}{\partial X}bdB$$

On applique le lemme au mouvement brownien géométrique

$$dX = \alpha X dt + \beta X dB$$

On s'intéresse à $G = ln(X)$.
On sait que :
$\frac{\partial G}{\partial X} = \frac{1}{S}$; $\frac{\partial^2 G}{\partial^2 X} = -\frac{1}{S^2}$; $\frac{\partial G}{\partial t} = 0$; $a = \alpha X$ et $b = \beta X$

$$dG = (\frac{\partial G}{\partial X}a + \frac{\partial G}{\partial t} + \frac{1}{2}\frac{\partial^2 G}{\partial^2 X}b^2)dt + \frac{\partial G}{\partial X}bdB$$

se simplifie en

$$dG = (\alpha - \frac{\beta^2}{2})dt + \beta dB$$

On constate que $G = ln(X)$ suit un processus brownien généralisé de drift constant $(\alpha - \beta^2/2)$ et un paramètre de variance constant β^2. Ainsi la variation de $ln(X)$ entre 0 et T suit une loi normale de paramètres $(\alpha - \beta^2/2)T$ et de variance $\beta^2 T$. X est donc distribué de façon log-normale.

8.4.2 Résolution du problème

Nous supposons que la courbe $C(t)$ suit un processus brownien géométrique, ce qui signifie :

$$C(t) = C(t_a)\exp(\rho t + \beta B_t^1),$$

Avec B^1 un processus brownien standard uni-dimensionnel. En outre, $e(t)$ est l'énergie consommée, correspondant à un instant t (la valeur de l'intégrale de $C(t)$ à l'instant t), par conséquent :

$$de(t) = C(t)dt \text{ ou bien } e(t_b) - e(t_a) = \int_{t_a}^{t_b} C(t)\,dt.$$

Rappelons que $C(t_a)$, $C(t_b)$, $e(t_a)$ et $e(t_b)$ sont des valeurs connues.

Il est difficile de simuler de tels processus car le comportement stochastique est antonymique avec les conditions déterministes posées dans notre problématique. Nous

avons emprunté une idée de Delyon et al. [51] qui consiste à passer par des processus facilement simulables en prenant en compte les conditions limites, puis à utiliser le théorème de Girsanov.

Pour toute fonction f définie sur l'intervalle $[t_a, t_b]$, la solution est donnée par l'expression de l'espérance suivante :

$$E\left[f(C,e)\,\middle|\,\begin{pmatrix}C(t_a)\\e(t_a)\end{pmatrix},\begin{pmatrix}C(t_b)\\e(t_b)\end{pmatrix}\right] = \frac{E[f(Q^0,R^0)M(Q^0,R^0)]}{E[M(Q^0,R^0)]} \tag{8.4}$$

Avec M une fonction dépendant de deux processus Q^0 et R^0 facilement simulables.

8.4.3 Démonstration

Nous avons considéré pour la courbe $C(t)$ et son intégrale $e(t)$ à chaque instant t les deux processus suivants :

$$\begin{cases} dC(t) = \alpha C(t)dt + \beta C(t)dB_t^1 \\ de(t) = C(t)dt \end{cases}$$

On procède à un changement de variables en posant $a = 0$ et $b = T$. Il est plus facile d'introduire le processus $P(t) = \ln(C(t))$. Nous obtenons :

$$\begin{aligned} P(t) &= \ln(C(t)) = \rho t + \beta B^1(t) \\ E(t) &= e(t) = e(0) + \int_0^t e^{P(s)}\,ds + \epsilon\beta B^2(t), \end{aligned} \tag{8.5}$$

Avec $\rho = (\alpha - \beta^2/2)$ et B^2 un second mouvement brownien standard indépendant de B^1.
Nous introduisons ce bruit afin de prendre en compte les incertitudes dans la mesure de l'énergie. Il est également ajouté pour des raisons techniques, qui ne seront pas développées ici.

Théorème 8.4.3.1 *Soient les deux processus suivants :*

$$\begin{cases} dx_t = b(t,x_t)dt + \sigma(t,x_t)dB_t \\ dy_t = (b(t,y_t) + \sigma(t,y_t)h(t,y_t))dt + \sigma(t,y_t)dB_t \end{cases}$$

$y_0 = x_0$ *et* B_t *est un brownien standard. Pour toute fonction* f *définie sur*$[0,T]$*, on a :*

$$E[f(y,B)] = E[f(x,B)e^{\int_0^T h^*(t,x_t)dB_t - \frac{1}{2}\int_0^T |h(t,x_t)|^2 dt}]$$

Ce théorème est appelé théorème de Girsanov [51]. On l'applique à nos processus de départ (8.5), en posant :

$$\begin{cases} dP^0(t) = \beta dB^1(t) \\ dE^0(t) = \epsilon\beta dB^2(t) \end{cases}$$

$\Leftrightarrow$

$$\begin{cases} P^0(t) = P^0(0) + \beta B^1(t) \\ E^0(t) = E^0(0) + \epsilon\beta B^2(t) \end{cases} \tag{8.6}$$

Ce qui donne :

$$E[f(P,E)] = E[f(P^0,E^0)M(P^0,E^0)] \tag{8.7}$$

Avec :

$$M(P^0,E^0) = exp(\int_0^t \begin{pmatrix} \beta & 0 \\ 0 & \epsilon\beta \end{pmatrix}^{-1} \begin{pmatrix} \rho \\ e^{P^0(t)} \end{pmatrix} d(\begin{pmatrix} B^1(t) \\ B^2(t) \end{pmatrix}) - \frac{1}{2}\int_0^t \left[\begin{pmatrix} \beta & 0 \\ 0 & \epsilon\beta \end{pmatrix}^{-1} \begin{pmatrix} \rho \\ e^{P^0(t)} \end{pmatrix}\right] \left[\begin{pmatrix} \beta & 0 \\ 0 & \epsilon\beta \end{pmatrix}^{-1} \begin{pmatrix} \rho \\ e^{P^0(t)} \end{pmatrix}\right]^* ds$$

$$M(P^0,E^0) = exp(\frac{\rho}{\beta}B^1(t) + \frac{1}{\beta\epsilon}\int_0^t e^{P^0(s)}dB(s)^2 - \frac{1}{2\beta^2}\rho^2 t - \frac{1}{2\epsilon^2\beta^2}\int_0^t e^{2P^0(s)}ds)$$

D'après les équations 8.6,

$$\begin{cases} B^1(t) = \frac{1}{\beta}(P^0(t) - P^0(0)) \\ B^2(t) = \frac{1}{\epsilon\beta}(E^0(t) - E^0(0)) \end{cases}$$

Ce qui donne :

$$M(P^0,E^0) = exp(\frac{\rho}{\beta^2}(P^0(t)-P^0(0))+\frac{1}{\epsilon^2\beta^2}\int_0^t e^{P^0(s)}dE^0(s)-\frac{\rho^2}{2\beta^2}t-\frac{1}{2\epsilon^2\beta^2}\int_0^t e^{2P^0(s)}ds)$$

De plus, nous considérons les deux processus suivants :

$$\begin{aligned} Q^0(t) &= P^0(t) - \frac{t}{T}\left(P^0(T) - \ln C(T)\right) \\ R^0(t) &= E^0(t) - \frac{t}{T}\left(E^0(T) - e(T)\right). \end{aligned}$$

Il est immédiat de voir que :

$$\begin{aligned} Q^0(0) = P^0(0) = \ln C(0),\ Q^0(T) = \ln C(T), \\ R^0(0) = E^0(0) = e(0),\ R^0(T) = e(T). \end{aligned}$$

Ainsi les processus Q^0 et R^0 satisfont les conditions limites de notre problème. De plus, on sait que [51] :

$$E\left[f(P^0,E^0)\,\middle|\, \begin{array}{ll} P^0(T) = \ln C(T), & E^0(T) = e(T), \\ P^0(0) = \ln C(0), & E^0(0) = e(0) \end{array}\right] = E[f(Q^0,R^0)] \tag{8.8}$$

Ainsi, nous obtenons le résultat final suivant :

$$E\left[f(P,E)\,\middle|\, \begin{array}{ll} P(T) = \ln c(T), & E(T) = e(T), \\ P(0) = \ln C(0), & E(0) = e(0) \end{array}\right] = \frac{E[f(Q^0,R^0)M(Q^0,R^0)]}{E[M(Q^0,R^0)]}, \tag{8.9}$$

Preuve :
On sait que :

$$E[f(P,E)g(P(T),E(T))] = \int E[f(P,E)|P(T)=u,E(T)=v]g(u,v)d\mu_T(u,v)$$

D'après le théorème 8.4.3.1 :

$$E[f(P,E)g(P(T),E(T))] = E[f(P^0,E^0)g(P^0(T),E^0(T))M(P^0,E^0)]$$

$$E[f(P,E)g(P(T),E(T))] = \int E[f(P^0,E^0)M(P^0,E^0)|P^0(T)=u,E^0(T)=v]g(u,v)d\mu_T^0(u,v)$$

$$E[f(P,E)g(P(T),E(T))] =$$
$$\int E[f(P^0,E^0)M(P^0,E^0)|P^0(T)=u,E^0(T)=v]g(u,v)d\mu_T^0(u,v)\frac{d\mu_T(u,v)}{d\mu_T(u,v)} \quad (8.10)$$

En particulier, pour $f(P,E)=1$:

$$\int g(u,v)d\mu_T = \int E[M(P^0,E^0)|P^0(T)=u,E^0(T)=v]g(u,v)d\mu_T^0(u,v)\frac{d\mu_T(u,v)}{d\mu_T(u,v)}$$

Ce qui implique :

$$\frac{d\mu_T^0(u,v)}{d\mu_T(u,v)} = \frac{1}{E[M(P^0,E^0)|P^0(T)=u,E^0(T)=v]} \quad (8.11)$$

De 8.10 et 8.11, on déduit que :

$$E[f(P,E)|P(T)=u,E(T)=v] = \frac{E[F(P^0,E^0)M(P^0,E^0)|P^0(T)=u,E^0(T)=v]}{E[M(P^0,E^0)|P^0(T)=u,E^0(T)=v]}$$

8.4.4 Interpolation de la courbe et de sa variance

Nous ne pouvons toujours pas simuler un échantillon de la trajectoire des processus (P,E) sous les conditions limites. Toutefois, nous pouvons estimer l'espérance de toute fonction de la trajectoire. Afin d'estimer la courbe entre deux instants échantillonnés t_a et t_b, nous utilisons la fonction constante définie par $f(X)=X$. Ainsi, l'équation 8.4 devient

$$E\left[C(t)\,\middle|\,\begin{pmatrix}C(t_a)\\e(t_a)\end{pmatrix},\begin{pmatrix}C(t_b)\\e(t_b)\end{pmatrix}\right]$$

Voici la procédure suivie pour l'interpolation :
- Nous simulons K fois $\binom{P^0}{E^0}$ sur $[t_a,t_b]$ ($K=1000$ dans nos expérimentations)
- Pour chacune des K trajectoires :
  - Nous calculons $Q^0(t)$ et $R^0(t)$ en les points à interpoler entre t_a et t_b ;
  - Nous calculons $M(Q^0,R^0)$ (en approchant les intégrales par la méthode des trapèzes par exemple)

- Nous formons $S_i(t) = e^{Q^0(t)} M(Q^0, R^0)$
- Nous utilisons l'équation 8.9 afin de calculer l'estimation : $E[C(t)|\binom{C(t_a)}{e(t_a)}, \binom{C(t_b)}{e(t_b)}] = (\frac{1}{M}\sum_{i=1}^{M} S_i(t))/M(Q^0, R^0)$

La variance qui permet l'estimation de l'erreur est calculée en utilisant la fonction $f(X) = (X - E(X))^2$ car $Var(X) = E[X - E(X)^2]$. Nous appliquons cette fonction au résultat de l'équation (8.4) en procédant de la même façon que précédemment :

$$E\left[(C(t) - E(C(t))^2 \,\middle|\, \binom{C(t_a)}{e(t_a)}, \binom{C(t_b)}{e(t_b)}\right] \tag{8.12}$$

Afin de calculer les paramètres, nous utilisons les propriétés connues du mouvement brownien géométrique. En effet, nous savons que $ln(C(t))$ entre t_a et t_b suit une loi normale d'espérance $(\alpha - \beta^2/2)(t_b - t_a)$. L'estimation de α se fait donc grâce à l'expression :

$$\alpha = \beta^2/2 + \frac{1}{(t_b - t_a)}(ln(C(t_b)) - ln(C(t_a)))$$

Le paramètre β peut être estimé historiquement à partir des données du passé ou par des simulations. Par exemple, on connaît une approximation de la courbe à interpoler par un polynôme de second degré qui respecte la contrainte de l'énergie. On peut effectuer plusieurs tests avec des valeurs différentes de β jusqu'à obtenir celle qui s'approche le plus du polynôme. Dans nos expérimentations, nous avons fixé le paramètre β à 1.

8.4.5 Exemple

Nous considérons la même courbe de charge que celle utilisée dans l'exemple de l'approche naïve, à savoir une courbe avec les 5 valeurs :
$C = \{31.67, 30.33, 24.33, 23, 28\}$

Estimation entre deux valeurs sélectionnées

Nous rappelons que le pas d'échantillonnage est de 5, ce qui signifie que nous gardons les points $C(0) = 31.67$ et $C(4) = 28$, et que nous devons estimer les points intermédiaires en prenant en compte l'énergie consommée $E = 109.33$ ($e(t_a) = 0$ and $e(t_b) = E$). Nous avons considéré comme paramètres du brownien géométrique conditionné :
$\beta = 1, \quad \alpha = \beta^2/2 + \frac{(t_b - t_a)}{(ln(C(t_b)) - ln(C(t_a)))} = 0.62$
La figure 8.6 montre le résultat de l'interpolation (Inter.), il s'agit de la courbe en tiret. L'enveloppe de l'écart-type autour de cette interpolation (+/- un écart type) est représenté en pointillé (Env.). La courbe en continu (LC) correspond à la courbe originale C.

Avec la même machine utilisée dans le cas de l'approche naïve, nous avons estimé l'enveloppe et la courbe en 2.77 secondes sachant que nous avons simulé les courbes $K = 1000$ fois (tel que décrit dans la section 8.4.4) et que nous avons estimé 100

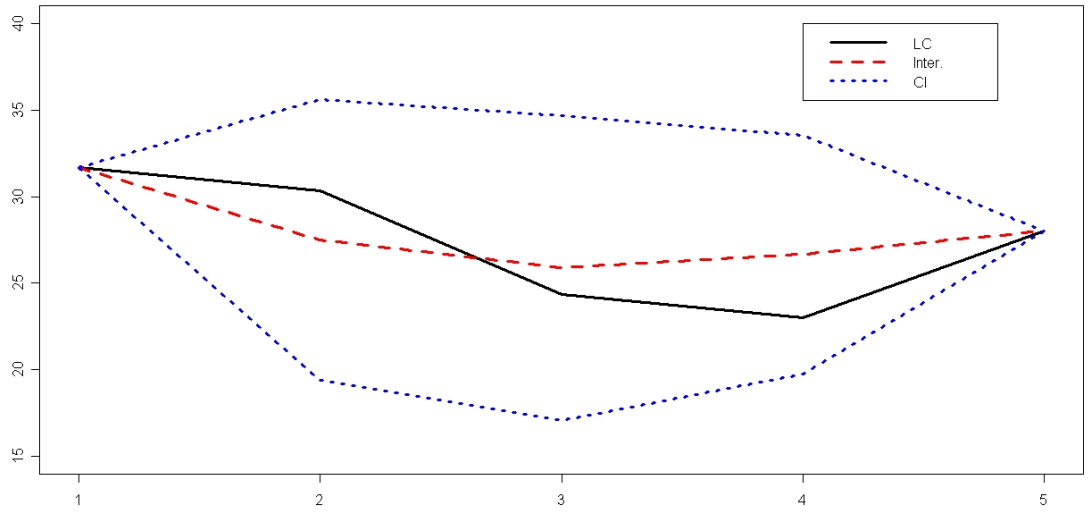

FIGURE 8.6 – Reconstruction de courbe de charge par l'approche stochastique entre deux points échantillonnés

éléments intermédiaires entre t_a et t_b. En effet, il est conseillé quand on fait une simulation avec un processus brownien de le faire sur un grand nombre de points intermédiaires afin de diminuer la variabilité entre deux points consécutifs. Ensuite, nous sélectionnons à partir des 100 points simulés ceux qui nous intéressent. Dans l'exemple donné ici, nous avons simulé 100 points entre $C(0)$ et $C(4)$ et nous avons sélectionné le 25ème, 50ème et le 75ème point pour reconstruire la courbe C.

Estimation de la courbe

Nous avons appliqué l'approche stochastique afin d'interpoler toute la courbe journalière en passant par les points sélectionnés. La figure 8.7 montre le résultat de l'interpolation en utilisant deux pas d'échantillonnage. Le premier pas d'échantillonnage de 5 est appliqué aux courbes de la figure 8.7(a) et le deuxième pas d'échantillonnage de 20 est appliqué aux courbes de la figure 8.7(b).

La partie gauche de la figure 8.7 montre un zoom des courbes de droite afin de décrire plus précisément la courbe originale et son interpolation grâce à l'approche stochastique. Les parties de droite de la figure montrent les intervalles de confiance entre chacune des valeurs collectées. Pour les deux taux d'échantillonnage, nous voyons clairement que l'interpolation par approche stochastique est très proche de la courbe originale.

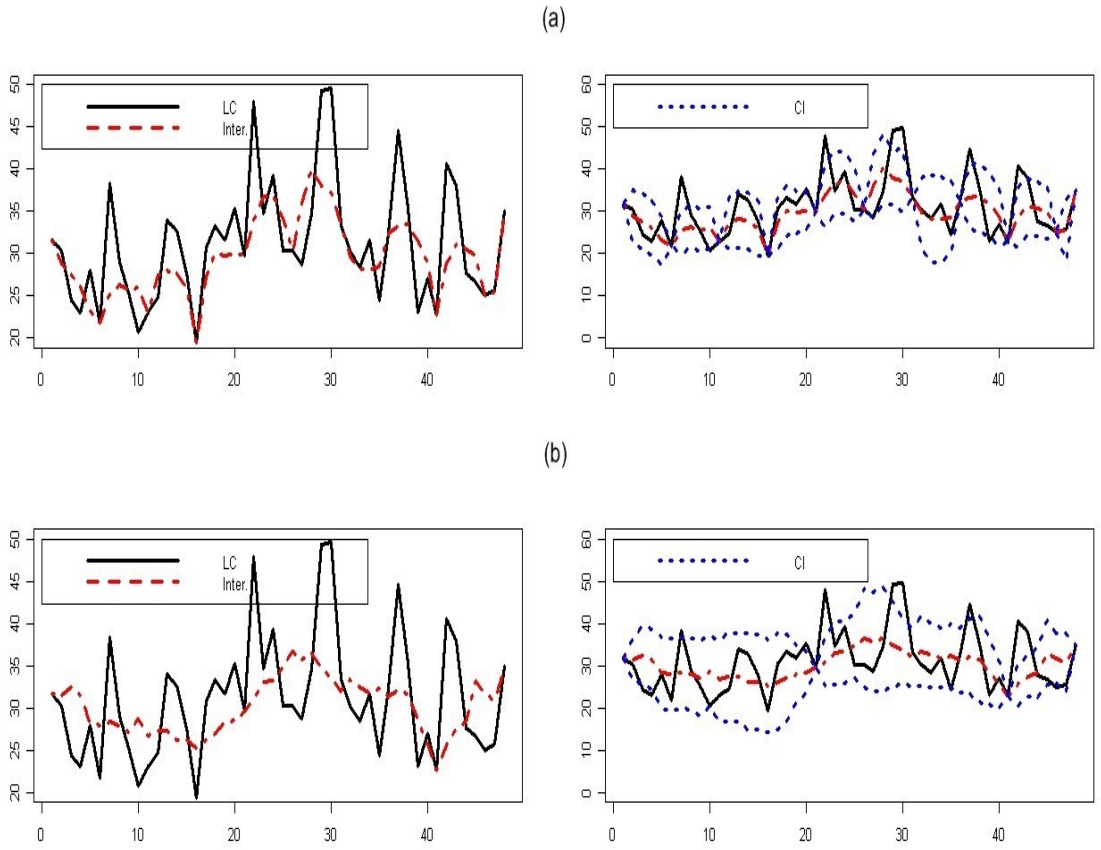

FIGURE 8.7 – Reconstruction de courbe de charge par l'approche stochastique entre deux points échantillonnés

8.5 Estimation sur petit domaine

Nous avons vu au chapitre 7 que l'échantillonnage spatial donne de meilleurs résultats dans le cas d'une estimation de la courbe globale de la population par rapport à l'échantillonnage temporel. Nous voulons comparer le comportement de ces deux approches pour des estimations sur des petits sous-ensembles de la population appelés aussi petits domaines. C'est l'objet des expérimentations menées dans la section 8.6 suivante, mais avant nous allons définir brièvement des notions liées aux estimations sur petit domaine. Pour de plus amples détails, le lecteur peut se référer au livre de J.N.K Rao [100].

Il arrive fréquemment que l'on souhaite établir des estimateurs à partir d'un échantillon afin d'inférer non pas sur toute la population mais seulement sur une partie de la population appelée domaine. Si on peut connaitre a priori le domaine d'intérêt, on choisit évidemment de ne tirer les individus que dans ce domaine. Malheureusement, ce n'est pas toujours le cas soit par manque d'information, soit parce qu'on s'intéresse à une étude simultanée de nombreux domaines, traités les uns après

les autres. Les individus sont donc le plus souvent tirés dans l'ensemble de la population sans que la constitution des domaines d'étude n'influe sur la méthode d'échantillonnage.

Appelons n la taille de l'échantillon global S, S_d représente l'échantillon qui recoupe le domaine d, n_d est la taille de S_d, et N_d la taille de la population du domaine. La variable d'intérêt est toujours la courbe $C(t)$ à l'instant t.

On considère un échantillon S tiré de manière aléatoire simple. On s'intéresse au vrai total C_d défni sur le domaine, son estimateur est donné comme pour Horvitz-Thompson par :

$$\hat{C}_d(t) = N_d \sum_{i=1}^{n_d} \frac{C_i(t)}{n_d} \tag{8.13}$$

Sa variance est calculée comme pour un estimateur d'Horvitz Thompson par :

$$\hat{Var}(\hat{C}_d(t)) = \frac{N_d(N_d - n_d)}{n_d(n_d - 1)} \sum_{j \in S_d} (C_j(t) - \frac{1}{n_d} \sum_{k \in S_d} C_k(t)) \tag{8.14}$$

Ces estimateurs sont vrais pour $n_d > 0$. De plus, la variance de l'estimateur varie comme l'inverse de la taille de l'échantillon dans le domaine. Cela rend difficile les estimations avec une précision acceptable sur de petits domaines où les tailles n_d sont très faibles, et où la précision peut donc devenir très mauvaise. Il existe une méthode permettant de contourner la difficulté. Il s'agit de prendre comme hypothèse que le domaine suit le même comportement que la population globale. On débouche sur une estimation dite synthétique du type :

$$\hat{C}_d(t) = N_d \sum_{i=1}^{n} \frac{C_i(t)}{n} \tag{8.15}$$

La précision devient meilleure puisque $\hat{Var}(\hat{C}_d(t))$ varie maintenant comme $\frac{1}{n}$ et non plus comme $\frac{1}{n_d}$ (avec n beaucoup plus grand que n_d). Toutefois, on se retrouve avec un risque de biais égal à $\sum_{i=1}^{N} \frac{C_i(t)}{N} - \sum_{i=1}^{N_d} \frac{C_i(t)}{N_d}$ si l'hypothèse de comportement formulée n'est pas vérifiée. Cette façon de faire revient donc à déplacer le risque d'une variance élevée vers le risque d'un biais.

Il existe d'autres estimateurs dans la littérature qui font généralement appel à une connaissance du comportement des individus dans la population ou à une connaissance de variables auxiliaires. Par exemple, un estimateur dit par prédiction considère les variables d'intérêts comme des variables aléatoires reliées à des variables auxiliaires par un modèle de type linéaire.

8.6 Etude expérimentale

Les expérimentations ont été effectuées sur le jeu de données de 1000 compteurs électriques présenté au chapitre 5. Afin d'obtenir les informations les plus précises sur la consommation électrique sans dépasser les limites de l'infrastructure de communication, nous appliquons notre approche temporelle de résumé de flux de données

distribués (en utilisant l'échantillonnage régulier) et nous la comparons à l'approche spatiale qui consiste en un échantillonnage aléatoire uniforme. Nous évaluons ces deux approches dans le cas des estimations sur petits domaines.

Nous utilisons des résumés construits par les deux approches spatiale et temporelle pour répondre à des requêtes typiques sur les courbes de charge. Nous citons parmi ces requêtes : la courbe de charge des clients habitant Paris, la courbe de charge globale des clients ayant souscrit à un tarif spécial, etc.

Nous n'avons pas de variables auxiliaires disponibles pour notre jeu de données afin de répondre aux requêtes ci-dessus. Par conséquent, nous avons sélectionné par simulations de Monte Carlo un certain nombre N_d de courbes à partir de la population. N_d est le nombre de courbes faisant partie du domaine d'intérêt. Nous estimons leur courbe de charge en utilisant seulement les résumés construits à partir des deux approches. Nous expérimentons plusieurs valeurs de N_d appartenant à l'ensemble $\{10, 20, 30, 50, 100, 200, 300, 500\}$ correspondant respectivement à 1%, 2%, 3%, 5%, 10%, 20%, 30% et 50% de la population.

Dans le cas de l'approche spatiale, nous considérons un échantillon de taille n construit aléatoirement et de façon uniforme. n est l'entier le plus grand tel que $p * n \leq s$, s étant le nombre total des données qu'on peut récupérer à partir des sources par période de temps. Nous nous intéressons au total $C_d(t)$ à chaque instant t défini uniquement sur les courbes du domaine d. Connaissant la taille de la sous-population N_d, l'estimateur de $C_d(t)$ est donné par l'équation (8.13). L'intervalle de confiance est calculé en utilisant l'estimation d'Horvitz-Thompson donnée par (8.3) et en remplaçant N par N_d et n par n_d pour ne prendre en compte que les courbes du domaine d'intérêt qui ont été tirées par échantillonnage spatial.

Quant à l'échantillonnage temporel, on détermine l'intervalle de confiance (Enveloppe) par l'intermédiaire de l'approche stochastique et l'approche naïve. Dans ce dernier cas, nous utilisons les programmes d'optimisation linéaire définis dans la section 8.3 basés sur l'estimation des pentes minimum et maximum à partir du passé. Nous avons utilisé ici les 100 premiers jours afin d'estimer la distribution des pentes.

Pour nos expérimentations, nous utilisons une précision de 68%. Ceci signifie que nous prenons +/- un écart-type dans le calcul de l'intervalle de confiance pour l'approche stochastique et pour l'échantillonne spatial, et nous choisissons les pentes minumum et maximum telle que la probabilité qu'une pente aléatoire soit dans l'intervalle $[\alpha_{min}, \alpha_{max}]$ est égale à 68% dans le cas de l'approche naive. Les figures 8.8 et 8.9 montrent les résultats des expérimentations avec les différentes valeurs de tailles des domaines. L'abscisse de la figure correspond aux tailles des domaines testées. Nous estimons la courbe de charge correspondant à un domaine en utilisant les deux approches, spatiale et temporelle. Nous avons procédé à 100 essais (simulations Monte Carlo) pour chaque N_d et nous avons dessiné la moyenne des erreurs dans les figures présentées ci-dessous. CI dans les figures 8.8 et 8.9 représente l'intervalle de confiance relatif (divisé par l'estimation de la courbe), et est calculé en additionnant les largeurs des enveloppes qu'on divise ensuite par l'estimation de la courbe globale du domaine. Le fait de sommer des enveloppes est très pessimiste en terme d'intervalle de confiance car ne prend pas en compte le foisonnement entre les différents capteurs. ErreurMoy correspond à la moyenne des erreurs absolues relatives entre la

vraie courbe agrégée et celle reconstruite à partir des résumés. Nous avons expérimenté différents taux de compression, c'est-à-dire, plusieurs tailles des échantillons spatial et temporel. Seuls deux exemples sont exposés ici car les autres donnent les mêmes résultats.

Un taux de compression de 50% (Figure 8.8) signifie que l'échantillon spatial consiste en $1000 \times 50\% = 500$ courbes et l'échantillon temporel consiste en $(144 \times 1000) \times 50\% = 72000$ éléments. Le haut de la figure 8.8 montre des courbes représentant l'intervalle de confiance relatif (avec une probabilité de 68%) avec chacune des méthodes de reconstruction des courbes. Nous observons que l'intervalle de confiance relatif en échantillonnant spatialement (courbe continue) est grand pour un très petit domaine et diminue quand celui-ci augmente de taille pour être inférieur aux intervalles de confiance avec les deux approches naive (courbe en tiret) et stochastique (courbe en pointillé). Nous observons aussi que les erreurs des estimations sont inférieures dans le cas de l'approche naive par rapport à l'approche stochastique et ceci pour toute taille du domaine. En effet, comme nous avions vu lors de l'exemple (cf. sous-section 8.3.3), l'approche naive donne une petite enveloppe pour des pas d'échantillonnage petits ce qui explique ce résultat (car les courbes sont échantillonnées en moyenne avec un pas d'échantillonnage de 2). Toutefois, on ne peut rien en déduire puisque nous ne connaissons pas précisément la probabilité de cet intervalle de confiance. Quant à l'approche stochastique, nous disposons d'une expression analytique qui permet de nous confirmer qu'avec une précision de 68%, l'approche stochastique donne un meilleur intervalle de confiance que l'échantillonnage aléatoire uniforme et ceci pour des petites tailles des domaines (ici $N_d < 150$). Par ailleurs, le bas de la figure 8.8 montre qu'en terme d'estimation de la courbe globale du domaine, c'est l'échantillonnage temporel (en particulier l'approche naive) qui l'emporte par rapport à l'approche spatiale.

La figure 8.9 montre les résultats avec un taux de compression de 10%. Dans ce cas, nous récupérons $1000 \times 10\% = 100$ courbes par l'approche spatiale et $(144 \times 1000) \times 10\% = 14400$ points par l'approche temporelle. En terme d'intervalle de confiance relatif, l'échantillonnage spatial (courbe continue) est meilleur que l'échantillonnage temporel (courbe en tiret pour l'approche naive et courbe en pointillé pour l'approche stochastique) pour des tailles de domaine supérieures à 150 courbes. Par contre, en terme d'erreur d'estimation empirique, c'est l'échantillonnage temporel qui donne les meilleurs résultats. Nous observons que les intervalles de confiance avec l'échantillonnage spatial sont calculés à partir d'une taille de domaine supérieure à 100 courbes. En effet, la taille de l'échantillon est petit et peut ne pas contenir aucune courbe du domaine d'intérêt. Ainsi la courbe totale estimée à partir du sous-échantillon du domaine est nulle. Ceci rend le calcul de l'intervalle de confiance relatif impossible. Nous utilisons dans ce cas l'estimation synthétique telle que nous avons vue à la section 8.5 et nous obtenons les résultats de la courbe en tiret-pointillé (courbe Synt).

Nous notons que chaque méthode a ses forces et ses faiblesses en fonction de la taille du domaine et du taux de compression. En terme d'estimation de la courbe globale, l'échantillonnage temporel avec l'approche naïve donne toujours de meilleurs résultats, suivi par l'échantillonnage temporel avec l'approche stochastique. Cette

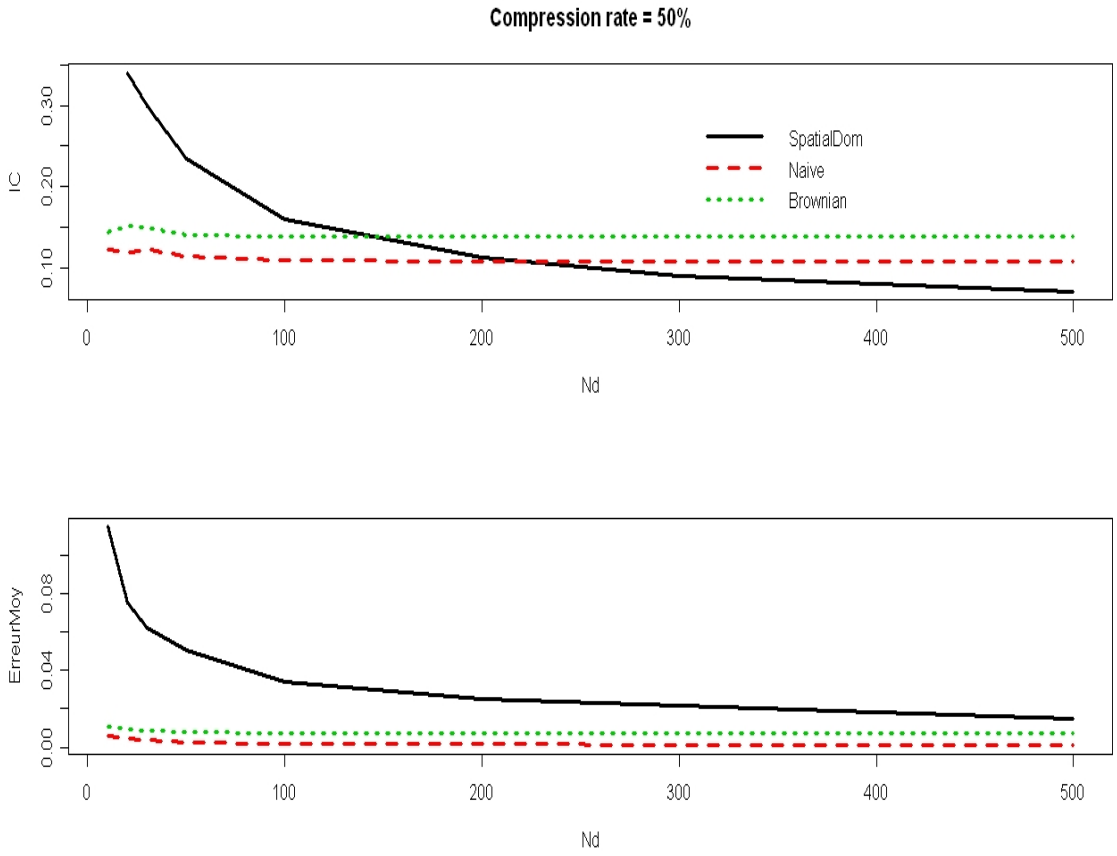

FIGURE 8.8 – Echantillonnage spatial Vs. échantillonnage temporel pour un taux de compression = 50%

dernière technique est meilleure en termes d'intervalle de confiance pour les grands taux de compression (grand échantillon) et des petites tailles de domaine. Enfin, l'échantillonnage spatial donne de bons résultats en termes d'intervalle de confiance en particulier pour les grands domaines.

8.7 Conclusion

Nous avons comparé notre méthode d'échantillonnage temporel visant à résumer les flux de données distribués à une approche spatiale qui est un échantillonnage aléatoire uniforme dans le cas des estimations sur petits domaines. Il existe des expressions de la théorie de sondage qui permettent d'estimer les réponses aux requêtes d'agrégation et qui fournissent des intervalles de confiance. Pour comparer l'approche spatiale à l'approche temporelle dans le cas des estimations sur petits domaines, il est nécessaire d'interpoler les flux de données résumés et d'estimer les erreurs de reconstruction à chaque instant. Nous avons proposé deux approches : la première dite

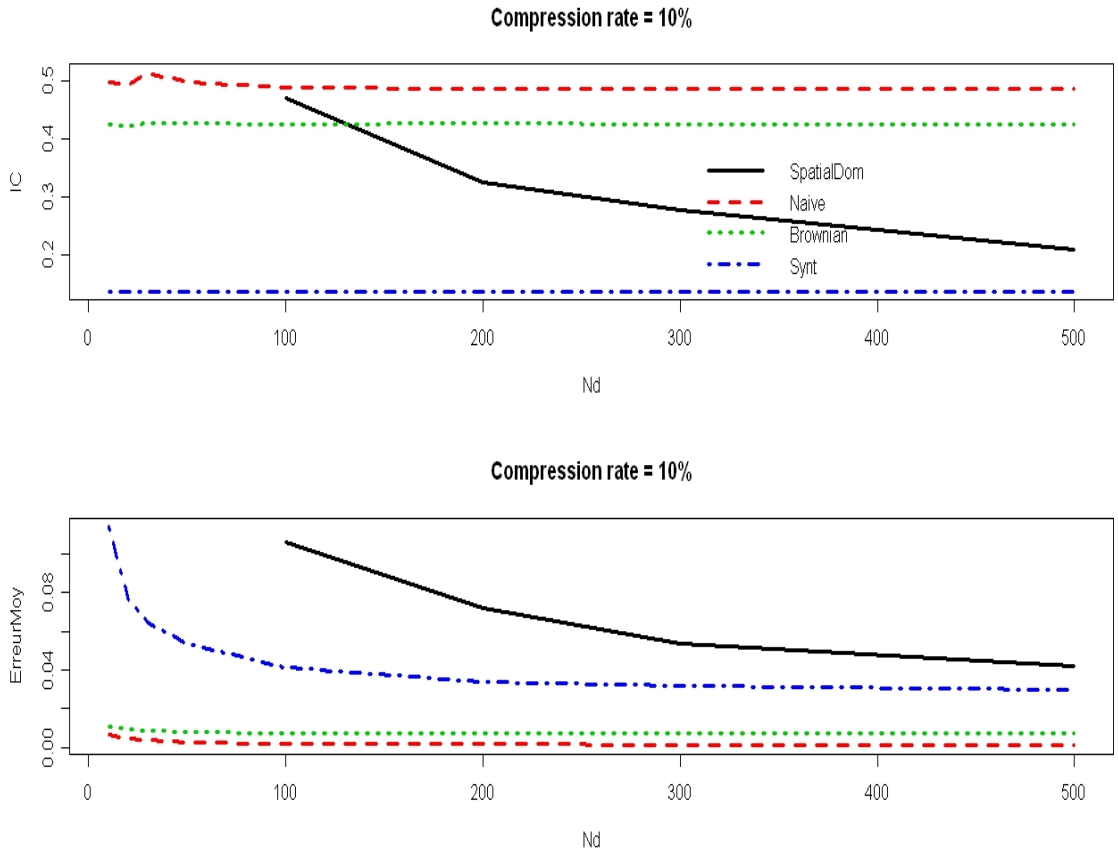

FIGURE 8.9 – Echantillonnage spatial Vs. échantillonnage temporel pour un taux de compression = 10%

naïve se base sur l'historique des courbes, et la seconde stochastique fait appel à un processus brownien afin d'interpoler la courbe. Nos expérimentations montrent l'intérêt de ces deux approches par rapport à l'approche spatiale afin d'estimer la courbe agrégée d'un petit domaine, et ceci pour toute taille de celui-ci. Néanmoins, l'échantillonnage spatial donne de meilleurs résultats en terme d'intervalle de confiance et en particulier pour les grands domaines.

Les expérimentations ont été effectuées sur un jeu de données de 1000 individus. Or, l'utilité du sondage réside dans la possibilité de résumer une population avec une taille très grande. En effet, la précision du sondage ne dépend que de la taille de l'échantillon et non du taux de sondage (donc du taux de dompression). Alors que la précision d'un échantillon temporel dépend du taux de compression. De plus, échantillonner temporellement une grande population est plus coûteux en terme de temps de calcul. En effet, la complexité de l'approche naïve (qui est moins coûteuse que l'approche stochastique) basée sur la méthode du simplexe est polynomiale alors que la complexité d'un échantillon aléatoire uniforme est linéaire.

Dans le cas de l'approche naïve, nos expérimentations ont montré que l'estimation de la courbe agrégée est très bonne, mais que l'enveloppe fournie par la méthode est très pessimiste pour des grands taux de compression. Il sera nécessaire comme perspective de ce travail de trouver la relation entre le 'vrai' intervalle de confiance et l'enveloppe fournie par l'approche naive. Quant à la seconde approche d'interpolation appelée *approche stochastique* et basée sur une modélisation brownienne géométrique, elle ne prend pas en compte la contrainte sur la valeur maximale à ne pas dépasser. Ce qui revient à limiter la dérivée de la fonction stochastique du modèle. Or, par définition le mouvement brownien ne peut être dérivable, il est donc difficile de contraindre notre modèle stochastique à ne pas dépasser une valeur donnée. Néanmoins, le fait de ne pas introduire cette contrainte n'est pas très gênant, car la contrainte de l'intégrale implique un risque très faible (voire nul) de dépasser la valeur maximale.

Chapitre 9

Conclusion et perspectives

Dans cette thèse, nous avons présenté nos travaux portant sur les techniques de résumé des flux de données distribués. Il s'est agi de construire un historique des données provenant de multiples sources et sur un long horizon de temps sans nécessairement garder toutes les données des flux. Nos travaux trouvent leur application dans tout environnement faisant intervenir plusieurs capteurs (capteurs physiques, capteurs de sécurité, de communication, etc.) fournissant une grande quantité d'informations à analyser.

Après avoir approfondi l'état de l'art du domaine des flux de données c'est-à-dire les concepts liés à ce domaine très récent, les enjeux associés à la constitution d'un résumé, ainsi que les différentes techniques existantes, nous avons étudié les travaux appliqués aux flux de données provenant de capteurs distribués. A partir de cet état de l'art, nous avons fait ressortir deux approches pouvant naturellement être appliquées dans ce contexte : (1) résumer le contenu de chaque source de flux de données, c'est ce que nous avons appelé *échantillonnage temporel*, (2) constituer un panel à partir des sources de données, c'est ce que nous avons appelé *échantillonnage spatial*.

Nous avons ensuite proposé une technique d'échantillonnage temporel qui consiste en un outil générique et optimisé de construction de résumés à partir de flux de données distribués. Cet outil permet de distribuer de façon optimisée les fréquences d'échantillonnage à appliquer à chaque capteur en se basant sur l'historique des données. La mise en oeuvre de cet outil a permis de montrer que l'affectation des niveaux de résumé par optimisation linéaire permet de réduire significativement les erreurs de résumé par rapport à une méthode non adaptative, c'est-à-dire lorsque les ressources du système centralisé sont équitablement réparties entre les capteurs. Notre outil de résumé de flux de données distribués répond pleinement aux contraintes liées aux flux de données. En effet, le résumé, construit au fil de l'eau, prend en compte la dimension temporelle du flux et évolue dynamiquement avec les variations des données. Notre outil est très générique et peut s'adapter à toute sorte d'application. Sa généricité vient du fait que toutes les méthodes de résumé de flux de données individuel dont quelques unes ont été énumérées au chapitre 3 peuvent y être insérées. Ainsi, un utilisateur peut choisir les techniques de résumé de flux de données

individuel à intégrer dans l'outil ainsi que la métrique d'erreur qui s'adapte le mieux à l'application en question. Nous avons prouvé l'efficacité de notre outil sur un jeu de données réel concernant des flux de données provenant de compteurs électriques.

Afin de constituer un historique des comportements des capteurs, nous avons également étudié la technique des panels (échantillonnage spatial). L'application de cette technique repose sur le choix d'un échantillon représentatif qui peut se faire de différentes manières. La plus simple est le tirage aléatoire uniforme où tous les capteurs ont la même probabilité de faire partie de l'échantillon. Nous avons comparé cette technique d'échantillonnage spatial avec notre technique d'échantillonnage temporel dans un contexte d'estimation globale. En effet, nous avons estimé le total des données de la population à chaque instant en se basant sur les deux échantillons, spatial et temporel. Il en ressort que l'échantillonnage spatial est plus performant pour estimer la courbe globale de la population. Nous avons aussi proposé une approche hybride qui combine l'échantillonnage spatial et l'échantillonnage temporel afin de pouvoir piloter dynamiquement la fréquence de relève des données à partir d'une sélection de capteurs. Nous l'avons appelée *échantillonnage spatio-temporel*. Ainsi, il est possible de faire évoluer dynamiquement l'échantillon spatial en s'adaptant à l'évolution des données, à l'infrastructure et aux besoins d'analyse.

Afin de fournir une réponse à des requêtes d'agrégation telles que la somme ou la moyenne des données à chaque instant, nous avons utilisé les inférences statistiques dans le cas de l'échantillonnage spatial. Tandis que pour l'échantillonnage temporel, les courbes résumées ne sont pas toutes observées au même instant (courbes asynchrones), ce qui nécessite de les reconstruire pour pouvoir appliquer des calculs de moyennes et de totaux. Nous avons développé deux techniques d'interpolation pour reconstruire les courbes résumés en prenant en compte la connaissance de leurs intégrales. Nos techniques fournissent les erreurs de reconstruction qui dépendent de chaque instant et lèvent l'hypothèse que les erreurs suivent une loi normale d'espérance nulle et de variance constante. La première technique se base sur l'historique de la courbe, nous l'avons appelé *approche naive*. La seconde se base sur un modèle brownien géométrique afin de reconstruire la courbe résumée, nous l'avons appelé *approche stochastique*. Les expérimentations ont montré l'efficacité de nos techniques afin d'estimer la courbe agrégée d'un sous-ensemble de capteurs par rapport à un échantillonnage aléatoire uniforme.

Perspectives

La gestion de données à partir de capteurs distribués est devenue un domaine de recherche très actif. De nombreux défis restent encore à relever. En ce qui concerne notre travail, nous avons identifié plusieurs pistes intéressantes. Nous proposons d'explorer certaines d'entre elles en considérant deux axes principaux : le résumé temporel de flux de données distribués et l'interpolation des flux de données avec connaissance de leur intégrale.

Outil générique de résumé de flux de données distribués

Application

Une des perspectives la plus immédiate est d'évaluer notre outil de résumé de flux de données distribué en considérant d'autres applications réelles en plus de celle utilisée dans les expérimentations concernant les compteurs électriques. La seule condition pour ce faire est que les flux de données soient périodiques. Toutefois, une périodicité existe dans toute activité humaine et/ou faisant intervenir l'effet saisonnier (trafic routier, télécommunications, actifs boursiers, etc.).

Données multi-dimensionnelles

Nous avons fait l'hypothèse de travailler sur des données uni-dimensionnelles et régulières. Ceci n'est pas toujours le cas, par exemple, un compteur électrique peut envoyer la mesure de consommation électrique en puissance, la mesure en énergie et d'autres informations concernant les équipements du consommateur. Une perspective de notre travail est d'intégrer ces informations auxiliaires dans l'affectation des niveaux de résumé. Toujours à propos de la consommation électrique, un niveau de résumé dépendra non seulement de la courbe de charge mais aussi d'autres éléments comme par exemple les variations de température extérieure ou des types d'équipements dont dispose le client. Ceci permettra de réduire les erreurs dues au fait que nous nous basons sur la période précédente pour échantillonner la période courante, puisque nous prendrons en compte d'autres informations qui influencent le comportement de la courbe.

Réduction de la matrice d'optimisation

Une amélioration peut être introduite dans notre outil générique de résumé de flux de données distribués afin de réduire la matrice des erreurs impliquée dans le programme d'affectation des niveaux de résumé. Il s'agit de regrouper des capteurs entre eux selon les corrélations observées entre leurs flux de données. Ainsi les capteurs faisant partie du même groupe utiliseront un même niveau de résumé (même nombre d'éléments). Ceci permettra de réduire les coûts de traitements et aussi de répondre plus précisément aux requêtes concernant les capteurs d'un même groupe.

Echantillonnage adaptatif par capteur

Il peut arriver qu'un sous-ensemble de capteurs présente plus d'intérêt que d'autres, soit parce qu'ils ont plus d'importance du point de vue de la construction du résumé, soit parce qu'ils sont explicitement ciblés par l'analyse. A titre d'exemple, si on sait qu'un compteur électrique a une consommation très évolutive et donc nécessite d'être échantillonné à un pas très fin, on peut l'indiquer comme contrainte dans le module d'optimisation. L'idée est d'inclure dans le module d'optimisation une précision sur le résumé qu'on voudrait obtenir pour chaque capteur. Cette précision sera dynamique dans le temps selon les variations des données ainsi que les besoins de traitement.

Ainsi les niveaux de résumé dépendent du passé de la courbe et de la précision souhaitée pour les capteurs.

Un autre axe de recherche lié à l'échantillonnage adaptatif, est d'appliquer non plus un seul niveau de résumé par capteur et par période de temps, mais d'échantillonner les données d'un capteur à un pas variable dans le temps, selon la variabilité de son flux de données. A titre d'exemple, un compteur électrique ayant une variabilité quasi-constante se verrait affecter un seul niveau de résumé, alors qu'un autre compteur avec une variabilité forte, par exemple entre la journée et la nuit, se verrait affecter deux niveaux de résumé. Le problème à résoudre est alors de déterminer le nombre de niveaux de résumé à affecter à chaque capteur ainsi que la valeur de chaque niveau.

Optimisation stochastique

Afin d'évaluer notre approche, nous n'avons utilisé que des techniques déterministes dans notre étude expérimentale. Nous y avons été contraints car les niveaux de résumé sont affectés par optimisation linéaire. Nous aurions pu utiliser des techniques probabilistes tel que l'échantillonnage de Bernoulli qui est applicable aux flux de données. Néanmoins, il aurait fallu utiliser des programmes d'optimisation stochastique [81] pour l'affectation des niveaux de résumé.

Reconstruction du flux de données et estimation de l'erreur

Approche naïve

L'approche naïve que nous avons développée lors du chapitre 8 fournit un intervalle de confiance correspondant à l'ensemble des solutions qui sont calculés avec des programmes d'optimisation linéaire. Cette technique donne une très bonne estimation de la courbe interpolée mais son inconvénient est qu'on ne connait pas précisément la probabilité que l'interpolation soit à l'intérieur de cet intervalle de confiance. Il est évidemment important d'étudier la relation entre l'enveloppe des solutions fournie par l'approche naïve et le 'vrai' intervalle de confiance.

Autres techniques d'échantillonnage spatial

Nous avons comparé notre approche temporelle à l'approche spatiale qui consiste en un échantillonnage aléatoire uniforme. Or, d'autres techniques de sondage existent afin d'améliorer les estimations. Il est important, si on dispose de variables auxiliaires corrélées avec la variable d'intérêt, de les prendre en compte dans le tirage de l'échantillonnage ou dans le redressement des estimations. Une façon de procéder est d'intégrer les variables auxiliaires dans un échantillonnage stratifié. Une perspective sera donc d'expérimenter d'autres techniques d'échantillonnage et en particulier un échantillonnage stratifié avec comme variable de stratification le niveau de consommation électrique par exemple.

Taille de la population

Nous avons vu au chapitre 8 que l'échantillonnage temporel estime mieux la courbe globale sur un petit domaine comparé à l'approche spatiale. Cette comparaison a été faite sur la base du jeu de données dont nous disposions et qui est constitué de 1000 capteurs (une population de 1000 individus). La question que l'on peut se poser est la suivante : quel sera l'impact de la taille de la population dans les résultats de la comparaison ? Il est évident que le taux de compression sur une population qui change de taille affectera plus l'échantillonnage temporel que l'échantillonnage spatial. En effet, pour des échantillons dont la taille est petite comparée à celle de la population entière (donc taux de sondage faible), la précision de l'information recueillie dépend principalement de la taille de l'échantillon et non du taux de sondage. Ainsi par exemple, la précision est à peu près la même pour un échantillon de 1000 individus tirés parmi 1 000 000 que pour un échantillon de même taille tiré parmi 10 000 000. Une perspective serait de comparer les deux approches avec une population plus grande que celle dont on disposait lors de l'étude expérimentale.

Evolution de la population

Nous avons effectué nos expérimentations sur un échantillon spatial fixe qui constitue un panel des capteurs. En général, une partie du panel est renouvelée selon un intervalle de temps fixé, pour prendre en compte l'évolution de la population ou pour faire face aux non réponses ou à la lassitude des individus. Cependant, dans le cas des données provenant de capteurs, nous n'avons pas de phénomène de lassitude. De plus, le renouvellement du panel n'engendre pas de coût supplémentaire, puisque les capteurs sont déjà installés (compteurs installés chez les clients dans le cas particulier de la consommation électrique). Par conséquent, il est judicieux de faire évoluer dynamiquement le panel selon les variations des données observées, de l'évolution des besoins d'analyse (évolution des domaines d'intérêts) et des évolutions des données contractuelles (nouveaux clients, changement d'équipements, déménagements, etc.). L'évolution de l'échantillon n'a pas été étudiée dans le cadre de ce travail de thèse et constitue donc une perspective.

Enfin, pour promouvoir l'utilisation de notre travail, il serait utile de l'intégrer au sein d'un système de gestion de flux de données. Ceci est l'objectif du projet ANR MIDAS dont Télécom ParisTech fait partie.

Bibliographie

[1] D. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, et S. Zdonik. *Aurora : a new model and architecture for data stream management*, VLDB , 12(2) :120-139, 2003.

[2] D. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Yi. Xing, S. B. Zdonik ; *The Design of the Borealis Stream Processing Engine*, CIDR 2005 : 277-289

[3] T. Abdessalem, R. Chiky, G. Hébrail, J. L. Vitti. *Traitement de données de consommation électrique par un Système de Gestion de Flux de Données*, EGC 2007 : 521-532

[4] T. Abdessalem, L. Decreusefond, J. Moreira. *Probabilistic measurement of uncertainty in moving objects databases*, In BDA'2005, 21èmes Journées de Bases de Données Avancées, 2005.

[5] http ://modulogriffon.com/catalogue/Documents/chap_3.pdf

[6] C.C. Aggarwal. *Data Streams : Models and Algorithms*, (Springer) Ed. 2006.

[7] C.C. Aggarwal, J. Han, J. Wang, P. Yu. *A Framework for Clustering Evolving Data Streams*, VLDB Conference 2003.

[8] C. C. Aggarwal. *On Biased Reservoir Sampling in the Presence of Stream Evolution*, in Proceedings of the 32nd international Conference on Very Large Data Bases VLDB, vol. 32, p 607-618, 2006.

[9] http ://www.aleri.com/

[10] P. S. Almeida, C. Baquero, N. Preguiça et D. Hutchison. *Scalable Bloom Filters*, Information Processing Letters, 2007, Elsevier North-Holland, Inc., Amsterdam, The Netherlands.

[11] N. Alon , P. Gibbons, Y. Matias, M. Szegedy *Tracking Joins and Self Joins in Limited Storage*, ACM PODS Conference 1999.

[12] N. Alon , Y. Matias, M. Szegedy *The space complexity of approximating the frequency moments*, In Proc. of the 1996 Annual ACM Symp. on Theory of Computing, pages 20-29, 1996.

[13] M. Altinel, M.J. Franklin. *Efficient Filtering of XML Documents for Selective Dissemination of Information*, Proc. VLDB Conference, Cairo, September 2000.

[14] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, J. Widom. *STREAM : The Stanford Data Stream Management System*, Book chapter (2004).

[15] A. Arasu, S. Babu, et J. Widom. *The cql continuous query language : Semantic foundations and query execution*, Technical Report 2003-67, Stanford University, 2003.

[16] P.Ardilly. *Les techniques de sondage*, Editions Technip, 2006.

[17] B.Babcock, S.Babu, M.Datar, R.Motwani, J.Widom, *Models and issues in data stream systems*, ACM PODS 2002.

[18] B.Babcock, S.Babu, M.Datar, R.Motwani, et D. Thomas. *Operator scheduling in data stream systems*, The VLDB Journal 13, 4 (Dec. 2004), 333-353.

[19] B. Babcock, M. Datar, et R. Motwani. *Load Shedding Techniques for Data Stream Systems*, (short paper) In Proc. of the 2003 Workshop on Management and Processing of Data Streams (MPDS 2003), June 2003.

[20] B. Babcock, M. Datar, et R. Motwani. *Load shedding for aggregation queries over data streams*, In ICDE'04 : proceedings of the 20th international conference on data engineering, Washington, DC, USA, pp. 350-361. IEEE Computer Society, 2004.

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

[22] Brain Babcock, Mayur Datar, Rajeev Motwani et Liadan O'Callaghan. *Maintaining variance and k-medians over data stream windows*, PODS '03 : Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2003, San Diego, California.

[23] B. A. Bash, J. W. Byers, J. Considine. *Approximately uniform random sampling in sensor networks*, Proc. First workshop on Data management in Sensor Network (DMSN'04), 2004.

[24] C. Bauzer-Medeiros, O. Carles, F. Devuyst, G. Hébrail, B. Hugueney, M. Joliveau, G. Jomier, M. Manouvrier, Y. Naïja, G. Scemama, L. Steffan, (2006). *Vers un entrepôt de données pour le trafic routier*, 2ème Journée Francophone sur les Entrepôts de Données et l'Analyse en ligne (EDA 2006), Versailles (France), pp.119-137, Juin 2006, Revue des Nouvelles Technologies de l'Information - RNTI-B2.

[25] R. Bellman. *On the approximation of curves by line segments using dynamic programming*, Communications of the ACM, VOL.4, N.6, p.284, Juin 1961.

[26] H. Bloom. *Space/Time Trade-offs in Hash Coding with Allowable Errors*, Commun. ACM 13, 7 (Jul. 1970), 422-426.

[27] A. Broder, M. Charikar, A. Frieze, et M. Mitzenmacher. *Min-wise independent permutations*, Journal of Computer and System Sciences, 60 :630-659, 2000.

[28] L. O'Callaghan, A. Meyerson, R. Motwani, N. Mishra et S. Guha. *Streaming-Data Algorithms for High-Quality Clustering*, Proceedings of the 23rd International Conference on Data Engineering (ICDE 02), 2002, IEEE Computer Society.

[29] D. Carney, U. Çetintemel, A. Rasin, S. Zdonik, M. Cherniack, et M. Stonebraker. *Operator scheduling in a data stream manager*, In Proceedings of the 29th international Conference on Very Large Data Bases, 2003.

[30] G.C. Cawley, N.L.C. Talbot, R.J. Foxall, S.R. Dorling, D.P. Mandic *Heteroscedastic kernel ridge regression*, Neurocomputing Volume 57, March 2004, Pages 105-124.

[31] G.C. Cawley, N.L.C. Talbot, R.J. Foxall, S.R. Dorling, D.P. Mandic *Unbiased estimation of conditional variance in heteroscedastic kernel ridge regression*, Proceedings of the European Symposium on Arti6cial Neural Networks (ESANN-2003), Bruges, Belgium, April 2003.

[32] S. Chandrasekaran, O. Cooper, A. Deshpande, M. Franklin, J. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, et M. Shah. *TelegraphCQ : Continuous data flow processing for an uncertain world*, In CIDR, 2003.

[33] Fwu-Ranq Chang. *Stochastic optimization in continuous time*, Cambridge university Press, 2004.

[34] M. Charikar, K. Chen, et M. Farach-Colton. *Finding frequent items in data streams*, Theor. Comput. Sci., 312(1) :3-15, 2004.

[35] M. Charikar, L. O'Callaghan et R. Panigrahy. *Better streaming algorithms for clustering problems*, Proceedings of the thirty-fifth annual ACM symposium on Theory of Computing (STOC 03), 2003, San Diego, CA, USA.

[36] J. Chen, D. DeWitt, F. Tian, Y. Wang. *NiagaraCQ : A Scalable Continuous Query System for Internet Databases*, In Proc. ACM Int. Conf. on Management of Data, 2000, pp. 379-390.

[37] R. Chiky, B. Defude et G. Hébrail. *Définition et diffusion de signatures sémantiques dans les systèmes pair-à-pair*, EGC 2006 : 463-468

[38] D. Chu, A. Deshpande, J. M. Hellerstein, et W. Hong. *Approximate data collection in sensor networks using probabilistic models*, In ICDE. IEEE, 2006.

[39] J. Considine, F. Li, G. Kollios, et J. Byers. *Approximate aggregation techniques for sensor databases*, In ICDE, 2004.

[40] G. Cormode, M. N. Garofalakis, S. Muthukrishnan, et R. Rastogi. *Holistic aggregates in a networked world : distributed tracking of approximate quantiles*, In SIGMOD Int. Conf. on Management of Data, pages 25-36. ACM, 2005.

[41] G. Cormode et M. N. Garofalakis. *Streaming in a connected world : Querying and tracking distributed data streams*. Tutorial at VLDB 2006, SIGMOD 2007, EDBT 2008.

[42] G. Cormode, M. N. Garofalakis et D. Sacharidis. *Fast approximate wavelet tracking on streams*, Proceedings of the International Conference on Extending Database Technology (EDBT 06), 2006.

[43] G. Cormode et S. Muthukrishnan. *Improved Data Stream Summary : The Count-Min Sketch and its Applications*, Technical report, 2003.

[44] G. Cormode, S. Muthukrishnan *What's hot and what's not : Tracking most frequent items dynamically*, ACM PODS Conference 2003.

[45] G. Cormode, S. Muthukrishnan et Wei Zhuang. *Conquering the Divide : Continuous Clustering of Distributed Data Streams*, Proceedings of the 23rd International Conference on Data Engineering (ICDE 07), 2007.

[46] C. Cranor, Y. Gao, T. Johnson, V. Shkapenyuk et O. Spatscheck. *GigaScope : High Performance Network Monitoring with an SQL Interface*, In Proc. ACM Int. Conf. on Management of Data, 2002, p. 623.

[47] B. Csernel. *Résumé généraliste de flux de données*, Rapport de thèse 2008, ENST Paris.

[48] B. Csernel, F. Clérot et G. Hébrail. *StreamSamp : DataStream Clustering Over Tilted Windows Through Sampling*, Knowledge Discovery from Data Streams Workshop (ECML/PKDD), Berlin, September 2006.

[49] A. Deligiannakis, Y. Kotidis, et N. Roussopoulos. *Compressing historical information in sensor networks*, In SIGMOD Int. Conf. on Management of Data, pages 527-538. ACM, 2004.

[50] A. Deligiannakis et N. Roussopoulos. *Extended wavelets for multiple measures*, Proceedings of the 2003 ACM SIGMOD international conference on Management of Data (SIGMOD 03), 2003, San Diego, California, USA.

[51] B. Delyon et Y.Hu. *Simulation of conditioned diffusion and application to parameter estimation*, Stochastic Processes and their Applications, Volume 116, Issue 11, November 2006.

[52] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, et W. Hong. *Model-driven data acquisition in sensor networks*, In VLDB, pages 588-599. Morgan Kaufmann, 2004.

[53] A. Dessertaine. *Echantillon et flux de données - estimations de courbes de consommation électrique à partir de données fonctionnelles*, Colloque francophone sur les sondages 2007, Marseille.

[54] J. C. Deville. *Méthodes statistiques et numériques de l'analyse harmonique*, Annales de l'INSEE, p. 3-104, 1974.

[55] J. C. Deville. *Variance estimation for complex statistics and estimators : linearization and residual techniques*, Survey Methodology, 25, 2, 193-203, 1999.

[56] A. Dobra, M. Garofalakis, J. Gehrke, et R. Rastogi. *Processing complex aggregate queries over data streams*, In Proceedings of the 2002 SIGMOD Conference, pages 61-72, 2002.

[57] J.B. Durand, L. Bozzi, G. Celeux, C. Derquenne. *Analyse de courbes de consommation électrique par chaînes de Markov cachées*, Revue de Statistique Appliquée, 52 no. 4 (2004), p. 71-91.

[58] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. *Summary cache : a scalable wide-area Web cache sharing protocol*, IEEE/ACM Transactions on Networking, 8(3) :281-293.

[59] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani et J. D. Ullman. *Computing iceberg queries efficiently*, In Proc. of the 1998 Intl. Conf. on Very Large Data Bases, pages 299-310, 1998.

[60] R. Féraud, F. Clérot. *Vers l'échantillonnage d'un entrepôt de données*, 3èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 2007), Poitiers, Juin 2007.

[61] S. Ferrandiz. *Gestion de flux de données*, Note sur la gestion de flux de données, 2007D016, ENST Paris, Novembre 2007.

[62] P. Flajolet et G. N. Martin. *Probabilistic counting algorithms for data base applications*, volume 31, pages 182-209, 1985.

[63] M.Garofalakis, J.Gehrke, et R.Rastogi. *Querying and mining data streams : you only get one look*, ACM SIGMOD 2002 (tutorial).

[64] M. Garofalakis et A. Kumar. *Deterministic wavelet thresholding for maximum-error metrics*, Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS 04), 2004, Paris, France.

[65] J. Gehrke, F. Korn, et D. Srivastava. *On computing correlated aggregates over continual data streams*, In Proceedings of the 2001 SIGMOD Conference, pages 13-24, 2001.

[66] P. Gibbons et S. Tirthapura. Estimating simple functions on the union of data streams, In Proceedings of the 2001 SPAA Conference, pages 281- 291, 2001.

[67] P.B. Gibbons et Y. Matias. *New sampling-based summary statistics for improving approximate query answers*, In Proc. ACM SIGMOD Conf., 1998, Seattle, WA, USA.

[68] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan et M. Strauss. *Surfing Wavelets on Streams : One-Pass Summaries for Approximate Aggregate Queries*, Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 01), 2001, San Francisco, CA, USA.

[69] L.Golab, M.T.Ozsu. *Issues in Data Stream Management*, ACM SIGMOD Record, June 2003.

[70] M. Gondran, M. Minoux. *Graphes et algorithmes*, Eyrolles, Paris 1979.

[71] H. Gonzalez , J. Han, X. Li, D. Klabjan. *Warehousing and Analyzing Massive RFID Data Sets*, Proceedings of 22nd IEEE International Conference on Data Engineering (ICDE), 2006.

[72] S. Guha, N. Mishra, R. Motwani et L.O'Callaghan. *Clustering data streams*, Proceedings of the Annual Symposium on Foundations of Computer Science. IEEE, 2000.

[73] S. Guha, A. Meyerson, N. Mishra, R. Motwani et L. O'Callaghan. *Clustering Data Streams : Theory and Practice*, IEEE Transactions on Knowledge and Data Engineering, 2003.

[74] D. J. Hand. *Statistics and Data Mining : Intersecting Disciplines*, ACM SIGKDD Explorations, 1, 1, pp. 16-19, June 1999.

[75] D.J. Hand , H. Mannila , et P. Smyth. *Principles of data mining*, MIT Press. (2001).

[76] P. J. Haas. *Data-Stream Sampling : Basic Techniques and Results*, In Data Stream Management : Processing High Speed Data Streams. Springer-Verlag, 2007.

[77] G. Hartl et B. Li. Infer. *A bayesian inference approach towards energy efficient data collection in dense sensor networks*, In ICDCS, pages 371-380. IEEE Computer Society, 2005.

[78] B. Hugueney. *Représentation symbolique de courbes numériques*, Thèse de doctorat de l'Université Paris 6, 2003.

[79] A. Jain et E. Y. Chang. *Adaptive sampling for sensor networks*, In Proceeedings of the 1st international Workshop on Data Management For Sensor Networks : in Conjunction with VLDB 2004 Toronto, Canada.

[80] M. Jarke, M. Lenzerini, Y. Vassiliou , P. Vassiliadis. *Fundamentals of data warehouses*, 2nd edition, Springer. 2003.

[81] P. Kall et S. W. Wallace. *Stochastic Programming*, John Wiley & Sons, 1994.

[82] K. Karabuluta, A. Alkanb, A. S. Yilmaz *Long Term Energy Consumption Forecasting using Genetic Programming*. Mathematical and Computational Applications, Vol. 13, No. 2, pp. 71-80, 2008.

[83] P. Karras et N. Mamoulis, *One-pass wavelet synopses for maximum-error metrics*, Proceedings of the 31st international conference on Very Large Data Bases (VLDB 05), 2005, Trondheim, Norway.

[84] D. Kasprzyk, G.Duncan, G.Kalton, M.P.Singh. *Panel Surveys*, New York : J. W. Wiley and Sons, Inc, 1989.

[85] B. Kedem et K. Fokianos. *Regression Models for Time Series Analysis*. Wiley Series in Probability and Statistics. September 2002.

[86] E. Keogh, S. Chu, D. Hart et M. Pazzani. *An Online Algorithm for Segmenting Time Series*, In Proceedings of IEEE International Conference on Data Mining. pages 289-296, 2001.

[87] I. Lazaridis et S. Mehrotra. *Capturing sensor-generated time series with quality guarantees*, In ICDE, pages 429-442. IEEE, 2003.

[88] C. Liu, K. Wu, et M. Tsao. *Energy Efficient Information Collection with the ARIMA model in Wireless Sensor Networks*, Proceedings of IEEE GlobeCom 2005, St. Louis, Missouri, November, 2005.

[89] L. Liu, C. Pu et W. Tang. *Continual Queries for Internet Scale Event-Driven Information Delivery*, In IEEE Trans. Knowledge and Data Eng., 11(4) : 610-628, 1999.

[90] *http ://lpsolve.sourceforge.net/*

[91] G. Manku et R. Motwani. *Approximate frequency counts over data streams*, In Proceedings of the 28th VLDB Conference, pages 346-357, 2002.

[92] M. Misiti, Y. Misiti, G. Oppenheim et J.M. Poggi *Analyse en ondelettes de la consommation électrique en vue de l'agrégation et de la désagrégation de courbes pour la prévision de la consommation*, déc. 2005, Rapport de contrat de recherche.

[93] S. Muthukrishnan. *Data streams : algorithms and applications*, In Foundations and Trends in Theoretical Computer Science, Volume 1, Issue 2, August 2005.

[94] S. Nath, P. B. Gibbons, S. Seshan, et Z. R. Anderson. *Synopsis diffusion for robust aggregation in sensor networks*, in Proceedings of the 2nd international Conference on Embedded Networked Sensor Systems SenSys '04. ACM Press, 2004, New York, p 250-262.

[95] B. Nguyen, S. Abiteboul, G. Cobena, et M. Preda. *Monitoring XML data on the Web*, SIGMOD Rec. 30, 2 (Jun. 2001), 437-448.

[96] C. Ordonez. *Clustering binary data streams with K-means*, Proceedings of the 8th ACM SIGMOD Conference on Research issues in Data Mining and Knowledge Discovery (DMKD 03), 2003, San Diego, California.

[97] G. M. Phillips *Interpolation and Approximation by Polynomials*. Springer, first edition (April 8, 2003).

[98] V. Poosala, V. Ganti, Y. Ioannidis. *Approximate Query Answering using Histograms*, IEEE Data Eng. Bull, 1999.

[99] V. Poosala, Y. Ioannidis, P. Haas, E. Shekita. *Improved Histograms for Selectivity Estimation of Range Predicates*, ACM SIGMOD Conference 1996.

[100] J.N.K. Rao. *Small Area Estimation*. Wiley & Sons, Inc, 2005.

[101] J.O. Ramsay et B.W. Silverman. *Functional Data Analysis*, New-York : Springer-Verlag, 2005.

[102] Sakurai Y., Papadimitriou S., Faloutsos C., emphAutoLag : Automatic Discovery of Lag Correlations in Stream Data, ICDE 2005, Tokyo, Japan.

[103] E. J. Stollnitz, T. D. Derose et D. H. Salesin. *Wavelets for computer graphics : theory and applications*, Morgan Kaufmann Publishers Inc., 1996.

[104] M. Stonebraker, U. Çetintemel, et S. Zdonik. *The 8 requirements of real-time stream processing*, SIGMOD Rec. 34, 4 (Dec. 2005), 42-47.

[105] http ://www.streambase.com/

[106] Nesime Tatbul, Ugur Cetintemel, Stan Zdonik, Mitch Cherniack et Michael Stonebraker. *Load Shedding in a Data Stream Manager*, Proceedings of the 29th International Conference on Very Large Data Bases (VLDB), September, 2003.

[107] D. B. Terry, D. Goldberg, D. Nichols et B. M. Oki. *Continuous Queries over Append-Only Databases*, SIGMOD Conference 1992 : 321-330

[108] Y. Tillé. *Sampling Algorithms*, New-York : Springer-Verlag, 2006.

[109] Y. Tillé. *Théorie des sondages - Echantillonnage et estimation en populations finies - Cours et exercices avec solutions*, Dunod, 2001.

[110] http ://www.truviso.com/

[111] J. S. Vitter. *Random sampling with a reservoir*, ACM Trans. Math. Softw. Vol. 11, p 35-57, 1985.

[112] J. S. Vitter. *Faster methods for random sampling*, Comm ACM vol. 27, p 703-718, 1984.

[113] Sanford Weisberg. *Applied Linear Regression*. Third Edition published by Wiley/Interscience in 2005 (ISBN 0-471-66379-4).

[114] R. Willett, A. Martin, et R. Nowak. *Backcasting : adaptive sampling for sensor networks*, In Proceedings of the Third international Symposium on information Processing in Sensor Networks, Berkeley, California, 2004.

[115] X. Yu, S. Mehrotra, N. Venkatasubramanian, et W. Yang. *Approximate monitoring in wireless sensor networks*, Technical Report 382, UCI ICS, 2004.

[116] S. Zdonik, M. Stonebraker, M. Cherniack, U. Cetintemel, M. Balazinska, et H. Balakrishnan. *The Aurora and Medusa Projects*, Bulletin of the Technical Committe on Data Engineering IEEE Computer Society. March 2003. p.3-10. (Invited Paper).

[117] T. Zhang, R. Ramakrishnan et M. Livny. *BIRCH : An Efficient Data Clustering Method for Very Large Databases*, Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data (SIGMOD 96), 1996, Montreal, Quebec, Canada.

[118] Y. Zhu, D. Shasha. *StatStream : Statistical Monitoringvof Thousands of Data Streams in Real Time*, In Proc. 28th Int. Conf. on Very Large Data Bases, 2002, pp. 358-369.