Découverte de motifs à la demande dans une base de données distribuée

Lamine Diop[*,**], Cheikh Talibouya Diop[**], Arnaud Giacometti[*]
Dominique Li[*], Arnaud Soulet[*]

[*]Université de Tours, France
{arnaud.giacometti, dominique.li, arnaud.soulet}@univ-tours.fr
[**]Université Gaston Berger de Saint-Louis, Sénégal
{diop.lamine3,cheikh-talibouya.diop}@ugb.edu.sn

Résumé. De nombreuses applications s'appuient sur des bases de données distribuées. Pourtant, peu de méthodes de découverte de motifs ont été proposées pour les extraire sans centraliser les données. Il faut dire que cette centralisation est souvent moins coûteuse que la communication des motifs extraits. Pour contourner cette difficulté, cet article adopte une approche parcimonieuse en coûts de communication en fournissant à l'utilisateur des motifs à la demande. Plus précisément, nous proposons l'algorithme DDSAMPLING qui tire un motif dans une base de données distribuée proportionnellement à son intérêt. Nous démontrons son exactitude et analysons sa complexité en temps et en communication soulignant son efficacité. Enfin, une étude expérimentale montre sur plusieurs jeux de données la robustesse de DDSAMPLING face aux défaillances d'un site ou du réseau.

1 Introduction

De nombreuses applications requièrent un stockage et une manipulation de bases de données distribuées (Özsu et Valduriez, 2011). Le plus souvent, la centralisation des données est impossible à cause de contraintes légales ou techniques. Ainsi, Zhang et Zaki (2006) soulignent l'importance d'étendre la découverte de connaissances aux bases de données distribuées. Par exemple, les données du web sémantique sont réparties sur plusieurs triplestores accessibles uniquement via des requêtes SPARQL. Dans ce contexte, les propriétés décrivant une même entité (e.g., Paris) sont réparties sur plusieurs sites (e.g., Wikidata ou GeoNames). Cet article vise à extraire directement des motifs au sein de telles bases de données distribuées.

Peu de travaux de la littérature se sont intéressés à la découverte de motifs dans des bases de données distribuées (Cheung et al., 1996; Otey et al., 2003; Jin et Agrawal, 2006; Kum et al., 2006). Ces propositions se sont focalisées sur une extraction exhaustive des motifs en fusionnant les extractions réalisées localement sur chacun des sites. Malheureusement, le volume de données à transmettre entre les différents sites exige un coût de communication bien supérieur à la centralisation des données car les motifs sont nombreux par nature et les multiples extractions génèrent de multiples doublons. De plus, le coût de calcul de ces extractions parallèles est prohibitif même si des techniques d'élagage les diminuent sensiblement en contrepartie de

coûts de communication supplémentaires (Zhu et Wu, 2007; Zhu et al., 2011). Afin d'éviter le coût inéluctable d'une extraction exhaustive, nous proposons de fournir des motifs à la demande en bénéficiant de l'échantillonnage de motifs (Al Hasan et Zaki, 2009; Boley et al., 2011). L'utilisateur peut à tout moment obtenir un échantillon de motifs dont le tirage sera proportionnel à leur intérêt. En plus d'être peu coûteux à extraire, ils s'avèrent utiles dans de nombreuses tâches comme la classification (Boley et al., 2011), l'extraction d'outliers (Giacometti et Soulet, 2016) ou l'exploration interactive de données (Giacometti et Soulet, 2017).

Cet article cherche à tirer des motifs ensemblistes proportionnellement à une mesure d'intérêt dans une base de données distribuée en ayant les mêmes garanties que si toutes les données avaient été centralisées. Après avoir formalisé notre problème dans la section 3, nous proposons un algorithme générique appelé DDSAMPLING (Distributed Database Sampling) qui tire aléatoirement un motif au sein d'une base de données distribuée proportionnellement à une mesure d'intérêt (cf la section 4). De manière originale, il autorise une nouvelle classe de mesures d'intérêt (comprenant notamment la fréquence, l'aire et la contrainte de taille maximale). A notre connaissance, notre proposition est la première approche d'extraction de motifs qui autorise un partitionnement vertical ou hybride des données. Par ailleurs, nous démontrons l'exactitude de DDSAMPLING et étudions sa complexité. En deuxième lieu, dans la section 5, nous évaluons la qualité de DDSAMPLING face aux défaillances de communication ou aux pannes de certains sites sur différents jeux de données. De manière intéressante, la proportionnalité du tirage est peu altérée.

2 Travaux relatifs

Cet article concerne la découverte de motifs sur une base de données distribuée. Cette problématique diffère de la parallélisation d'algorithmes d'extraction où les calculs distribués sont opérés sur un jeu de données unique ou sciemment distribué pour accélérer les calculs.

Plusieurs approches de la littérature se sont intéressées à l'extraction de motifs fréquents sur une base de données distribuée. Cette tâche est complexe car quel que soit la fréquence exigée sur la base de données distribuée (fréquence globale), il n'est pas possible de contraindre la fréquence locale sur un fragment sans communiquer des informations entre sites. Dans ce cadre, Cheung et al. (1996) proposent le premier travail pour extraire tous les motifs globalement fréquents en identifiant les sites où les motifs sont les plus fréquents et en réduisant ainsi un peu les échanges. De manière plus drastique, Otey et al. (2003) proposent d'économiser les échanges en se limitant à la collection des motifs fréquents maximaux. Pour ne pas avoir à énumérer tous les motifs de chaque fragment et limiter les échanges, Jin et Agrawal (2006) imposent un seuil minimal de fréquence sur chaque fragment. A partir des différentes extractions locales, Kum et al. (2006) construisent une collection globale approchée des motifs fréquents. Un élagage centralisé proposé par Zhu et Wu (2007) repose sur la construction d'un arbre contenant pour chaque motif toutes ses occurrences (i.e., couples fragment/transaction), ce qui requièrt encore un volume d'échanges considérable. Plus récemment, Zhu et al. (2011) parviennent à mettre en oeuvre un élagage décentralisé au sein de l'extraction de chaque fragment en échangeant des filtres de Bloom. Cette approche réduit significativement les temps de calculs mais le coût des communications amoindri demeure important. En effet, le volume de motifs extraits génère invariablement un coût de communications énorme bien supérieur à celui de la centralisation des données. En outre, toutes ces approches d'extraction de motifs

fréquents se restreignent à un partitionnement horizontal des données, une même transaction ne pouvant pas être distribuée sur deux fragments distincts. Enfin, les propositions de l'état de l'art requièrent d'avoir une capacité de calcul importante sur chaque fragment ce qui n'est pas toujours possible. Par exemple, le web sémantique offre un accès à des données distribuées via une requête SPARQL mais il n'est pas possible d'y exécuter une routine d'extraction. Ainsi, pour éviter des coûts de calcul sur chaque fragment et des coûts de communications prohibitifs, nous optons pour une extraction de motifs à la demande.

L'extraction de motifs à la demande consiste à extraire en un temps très court un motif sans avoir pré-calculé auparavant une énorme collection de motifs. Par exemple, de nombreuses méthodes heuristiques (Dietterich et Michalski, 1983) permettent d'induire rapidement des règles dans un contexte supervisé. Plus récemment, Al Hasan et Zaki (2009); Boley et al. (2011) ont proposé des méthodes d'échantillonnage de motifs qui extraient instantanément des motifs. Ces méthodes visent à tirer des motifs avec une distribution de probabilité proportionnelle à leur intérêt. Etant dans un contexte non-supervisé, notre article s'inscrit dans cette dernière direction. De manière plus précise, les techniques d'échantillonnage exactes se répartissent en deux grandes catégories : les méthodes stochastiques (Al Hasan et Zaki, 2009) et les méthodes en plusieurs étapes (Boley et al., 2011). Afin de marcher aléatoirement d'un motif X à un autre, les méthodes stochastiques requièrent d'accéder à l'intérêt global de tous les motifs voisins à X. Par exemple, dans le cas de la fréquence, il faudrait connaître la fréquence globale de tous les sous-ensembles et de tous les sur-ensembles de X, ce qui occasionnerait de nombreux coûts de communication. Pour cette raison, nous préférons adopter une procédure aléatoire de tirage en plusieurs étapes. Il est vrai que cette technique a déjà été déclinée pour plusieurs mesures d'intérêt (e.g., support, aire (Boley et al., 2011) ou mesure d'exception (Moens et Boley, 2014)) et plusieurs types de données (e.g., données numériques (Giacometti et Soulet, 2018) ou séquentielles (Diop et al., 2018)). Néanmoins, le contexte des bases de données distribuées constitue un challenge orthogonal. Par simplicité, nous nous limitons dans cet article aux motifs ensemblistes, mais notre approche considère différentes mesures d'intérêt.

3 Formalisation du problème

3.1 Motifs ensemblistes et bases de données distribuées

Soit $\mathcal{I}$ un ensemble fini de littéraux nommés items. Un itemset ou motif X est un sous-ensemble non vide de $\mathcal{I}$. La taille de l'itemset X, notée $|X|$, est sa cardinalité. L'ensemble de tous les itemsets définis sur $\mathcal{I}$, dénoté par $\mathcal{L}$, s'appelle le langage. Dans ce contexte, une base de données $\mathcal{D}$ est un ensemble de couples définis sur $\mathbb{N}\times\mathcal{L}$: $\mathcal{D} = \{(j, X) : j \in \mathbb{N} \wedge X \in \mathcal{L}\backslash\{\emptyset\}\}$. $\mathcal{D}[j]$ correspond à l'itemset X décrivant la transaction j si $(j, X) \in \mathcal{D}$, alors que $\mathcal{D}[j] = \emptyset$ si aucune transaction d'identifiant j appartient à $\mathcal{D}$. $|\mathcal{D}|$ est le nombre de transactions de $\mathcal{D}$ et $||\mathcal{D}|| = \sum_{j\in\mathbb{N}} |\mathcal{D}[j]|$ définit la taille de $\mathcal{D}$. Par exemple, la figure 1 présente plusieurs bases de données contenant entre 2 et 5 transactions décrites par les items $A, \ldots, G$. Pour la base de données $\mathcal{D}$, on a $\mathcal{D}[2] = ADFG$. Enfin, on a $|\mathcal{D}| = 5$ et $||\mathcal{D}|| = 3 + 4 + 3 + 4 + 3 = 17$.

Intuitivement, une base de données distribuée est juste un ensemble de bases de données où les transactions de chaque fragment ne se chevauchent pas :

Définition 1 (Base de données distribuée/centralisée) *Une base de données distribuée $\mathcal{P} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$ est un ensemble de bases de données (appelées fragments) tel que pour tout*

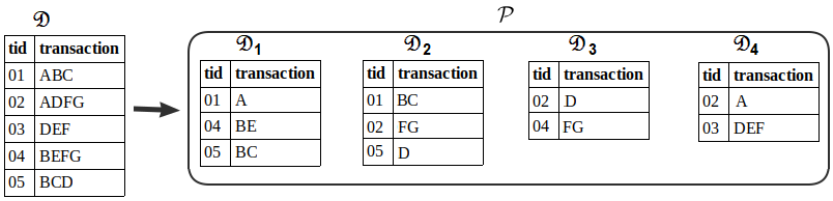

FIG. 1: Exemple d'une base de données distribuée $\mathcal{P} = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4\}$

identifiant de transaction $j \in \mathbb{N}$, on a $\mathcal{D}_k[j] \cap \mathcal{D}_l[j] = \emptyset$ pour tous les fragments $\mathcal{D}_k$ et $\mathcal{D}_l$ où $k \neq l$. La base de données centralisée $\mathcal{D}$ correspondant à $\{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$ est la fusion de toutes les transactions de chacun des fragments : $\mathcal{D} = \{(j, \mathcal{D}_1[j] \cup \cdots \cup \mathcal{D}_K[j]) \in \mathbb{N} \times \mathcal{L} \ : \ \exists k \in [1..K]$ *tel que* $\mathcal{D}_k[j] \neq \emptyset\}$.

Dans la suite, sauf indication contraire, $\mathcal{P} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$ désigne une base de données distribuée de K fragments et $\mathcal{D}$ correspond à sa base de données centralisée. On dit que la base de données distribuée $\mathcal{P} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$ est un partitionnement de la base de données centralisée $\mathcal{D}$. Par exemple, dans la figure 1, on constate que la base de données distribuée $\mathcal{P} = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4\}$ est un partitionnement de la base de données centralisée $\mathcal{D}$. Dans ce cas, la transaction 2 est répartie sur les fragments $\mathcal{D}_2$, $\mathcal{D}_3$ et $\mathcal{D}_4$. De manière plus générale, suivant la répartition des données sur les fragments, on distingue deux partitionnements particuliers de $\mathcal{D}$. Si chaque transaction est contenue sur un seul fragment, on parle de partitionnement horizontal. Si chaque item est contenu sur un seul fragment, on parle de partitionnement vertical. Un partitionnement hybride est un partitionnement qui n'est ni horizontal, ni vertical (e.g., partitionnement de la figure 1). Par exemple, dans le cas des données du web sémantique, une entité est décrite sur plusieurs triplestores. Dans ce cas, il est pertinent d'extraire des motifs sur l'ensemble de la base de données distribuée pour rechercher des corrélations entre les propriétés de triplestores distincts.

Pour finir, comme indiqué auparavant, nous considérons un environnement avec une base de données distribuée où il est seulement possible de demander à un fragment k, la taille d'une transaction d'identifiant j et l'item à la position i d'une transaction j :

1. La requête $\texttt{sizeOf}(j, \mathcal{D}_k)$ retourne la taille de la transaction d'identifiant j du fragment $\mathcal{D}_k$ i.e., $\texttt{sizeOf}(j, \mathcal{D}_k) = |\mathcal{D}_k[j]|$. Dans notre exemple, on a $\texttt{sizeOf}(4, \mathcal{D}_1) = 2$ car $|\mathcal{D}_1[4]| = |BE| = 2$.
2. La requête $\texttt{itemAt}(i, j, \mathcal{D}_k)$ retourne le $i$ème item de la transaction d'identifiant j du fragment $\mathcal{D}_k$ en supposant un ordre arbitraire sur les items de $\mathcal{I}$. Par exemple, avec l'ordre lexicographique sur les items, nous avons $\texttt{itemAt}(2, 4, \mathcal{D}_1) = E$.

Nous formalisons notre approche sur ces opérations élémentaires dans un soucis de généricité même si en pratique des opérations plus complexes sont possibles.

3.2 Echantillonnage de motifs sur une base de données distribuée

En découverte de motifs, il est primordial d'évaluer l'intérêt d'un motif pour savoir s'il est pertinent pour l'utilisateur. La plus populaire des mesures est probablement le sup-

port défini comme la proportion de transactions de la base de données $\mathcal{D}$ contenant le motif X : $supp(X, \mathcal{D}) = |\{(j, T) \in \mathcal{D} : X \subseteq T\}|/|\mathcal{D}|$. Avec l'exemple de la figure 1, on a $supp(DF, \mathcal{D}) = 2/5$ car le motif DF apparaît dans les transactions 2 et 3. Il est courant d'associer aussi une utilité à un motif en fonction de sa taille comme c'est le cas avec la mesure d'aire : $supp(X, \mathcal{D}) \times |X| \times |\mathcal{D}|$. De ce fait, nous nous intéressons à une famille de mesures d'intérêt de la forme $supp(X, \mathcal{D}) \times u(X)$ où u est une fonction d'utilité fondée sur la taille :

Définition 2 (Utilité fondée sur la taille) *Une fonction d'utilité u est dite fondée sur la taille ssi il existe une fonction $f_u : \mathbb{N} \to \mathbb{R}$ telle que $u(X) = f_u(|X|)$ pour tout motif X.*

Dans la suite, et par simplicité, toute utilité d'un motif réfère à une fonction d'utilité fondée sur la taille. Par l'exemple, l'utilité $u_{area} = |X| \times |\mathcal{D}|$ permettra de considérer la mesure d'aire $supp(X, \mathcal{D}) \times u_{area}(X)$ et dans ce cas, on a $f_{u_{area}}(\ell) = \ell \times |\mathcal{D}|$. L'utilité $u_{\leq M}$ définit par 1 si $|X| \leq M$ et 0 sinon imposera une contrainte de taille maximale car avec la mesure induite $supp(X, \mathcal{D}) \times u_{\leq M}(X)$, un motif dont la cardinalité est strictement supérieure à M sera jugé inutile (quel que soit sa fréquence). Notons qu'avec la fonction d'utilité $u_{freq}(X) = 1$, nous considèrerons tout simplement le support comme mesure d'intérêt. Enfin, de manière intéressante, notons que le produit ou la somme de deux fonctions d'utilité fondées sur la taille reste une fonction d'utilité fondée sur la taille. Ainsi, $u_{area} \times u_{\leq M}$ définira comme utile un motif avec une grande aire tant que sa cardinalité n'excède pas le seuil M.

L'échantillonnage de motifs vise à tirer aléatoirement un motif X parmi le langage $\mathcal{L}$ avec une probabilité $\pi = m(X)/Z$ où m est une mesure d'intérêt et Z une constante de normalisation. Formellement, nous noterons ce tirage de la manière suivante : $X \sim \pi(\mathcal{L})$. Dans la suite, $\mathcal{L}$ désigne le langage commun à tous les fragments de la base de données distribuée.

Soient une base de données distribuée $\mathcal{P} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ et une fonction d'utilité fondée sur la taille u. Notre objectif est de tirer un motif X de $\mathcal{L}$ proportionnellement à $supp(X, \mathcal{D}) \times u(X)$, i.e. $X \sim \pi(\mathcal{L})$ avec $\pi(X) = supp(X, \mathcal{D}) \times u(X)/Z$, où $\mathcal{D}$ est la base de données centralisée de $\mathcal{P}$ et en utilisant uniquement des requêtes `sizeOf` *et* `itemAt`.

4 Méthode d'échantillonnage par fragments

4.1 Approche naïve par centralisation des données

Une première solution au problème défini ci-dessus serait de construire une base de données centralisée $\mathcal{D}$ en effectuant des requêtes `itemAt` de manière intensive ; puis, d'appliquer un algorithme d'échantillonnage classique sur $\mathcal{D}$. Dans cette persective, nous reformulons l'algorithme de tirage en deux étapes de Boley et al. (2011) dédié à la mesure d'aire pour toute mesure d'intérêt de la forme $supp(X, \mathcal{D}) \times u(X)$ où u est une fonction d'utilité fondée sur la taille. L'algorithme 1 s'applique sur une base de données centralisée. *Etape 1 :* Nous calculons la pondération de chaque transaction j de $\mathcal{D}$ en sommant l'utilité $u(X)$ de chaque sous-ensemble de X de $\mathcal{D}[j]$ (ligne 1). Ensuite, un identifiant de transaction j est tiré au hasard proportionnellement à son poids $\omega(j)$ (ligne 2). *Etape 2 :* La ligne 3 détaille le poids $\omega(j)$ afin de connaître pour chaque longueur ℓ, le poids de tous les motifs de $\mathcal{D}[j]$ qui ont exactement cette longueur ℓ. La ligne 4 utilise cette distribution pour tirer au hasard une longueur ℓ proportionnellement à $\omega_\ell(j)$. Enfin, comme tous les motifs de longueur ℓ ont le même poids, il suffit de retourner un motif tiré uniformément parmi ceux qui ont une longueur ℓ (ligne 5).

Algorithm 1 Tirage d'un motif dans une base de données centralisée (Boley et al., 2011)

Input: Une base de données centralisée $\mathcal{D}$ et une fonction d'utilité u
Output: Un itemset tiré aléatoirement $X \sim \pi(\mathcal{L})$ où $\pi(X) = supp(X, \mathcal{D}) \times u(X)$
// Etape 1 : tirage centralisé d'un identifiant de transaction
Soient les poids ω définis par $\omega(j) := \sum_{X \subseteq \mathcal{D}[j]} u(X)$ pour tout $j \in \mathbb{N}$
Tirer une transaction j proportionnellement à $\omega : j \sim \omega(\mathbb{N})$
// Etape 2 : tirage centralisé d'un itemset
Soient les poids définis par $\omega_\ell(j) := \sum_{X \subseteq \mathcal{D}[j] \wedge |X|=\ell} u(X)$ pour tout $\ell \in [0..|\mathcal{D}[j]|]$
Tirer un entier ℓ proportionnellement à $\omega_\ell(j) : \ell \sim \omega_{[0..|\mathcal{D}[j]|]}(j)$
return un itemset de taille ℓ de $\mathcal{D}[j] : X \sim unif(\{X \subseteq \mathcal{D}[j] \ : \ |X| = \ell\})$

Bien sûr, la centralisation des données dont le coût en communication requiert $O(||\mathcal{D}||)$ requêtes est bien trop élevé. Il est donc préférable de parvenir à effectuer un échantillonnage directement sur la base de données distribuée. Pour cela, l'idée clé de notre approche est de conserver le tirage d'un identifiant de transaction j proportionnellement à $\omega(j)$ et d'une longueur ℓ proportionnellement à $\omega_\ell(j)$, mais surtout de tirer indépendamment ℓ items dans les différents fragments. Malheureusement, cela implique de relever deux challenges. Premièrement, il faut parvenir à décentraliser le calcul des poids $\omega_\ell(j)$ sans échanger d'items (pour éviter des coûts de communication). Deuxièmement, il faut parvenir à tirer plusieurs itemsets sur différents fragments de sorte à émuler un tirage uniforme sur $\mathcal{D}[j]$.

4.2 Approche sans centralisation des données : DDSAMPLING

Algorithm 2 DDSAMPLING

Input: Une base de données distribuée $\mathcal{P} = \{\mathcal{D}_1, \cdots, \mathcal{D}_K\}$ et une utilité fondée sur la taille u
Output: Un itemset tiré aléatoirement $X \sim \pi(\mathcal{L})$ où $\pi(X) = supp(X, \mathcal{D}) \times u(X)$
// Phase de prétraitement
Soit $\mathbb{M}$ la matrice définie par $\mathbb{M}[j][k] := \texttt{sizeOf}(j, \mathcal{D}_k)$ pour tout $j \in [1..|\mathcal{D}|]$ et $k \in [1..K]$
Soient les poids définis par $\omega(j) := \sum_{\ell=0}^{\widetilde{\mathbb{M}}_j} \binom{\widetilde{\mathbb{M}}_j}{\ell} \times f_u(\ell)$ pour tout $j \in [1, |\mathcal{D}|]$
// Phase de tirage
// Etape 1 : tirage décentralisé d'un identifiant de transaction
Tirer un indice j compris entre 1 et $|\mathcal{D}|$ proportionnellement à $\omega : j \sim \omega(\mathcal{D})$
// Etape 2 : tirage fragmenté d'un itemset
Soient les poids définis par $\omega_\ell(j) := \binom{\widetilde{\mathbb{M}}_j}{\ell} \times f_u(\ell)$ pour tout $\ell \in [0..\widetilde{\mathbb{M}}_j]$
Tirer un entier ℓ proportionnellement à $\omega_\ell(j) : \ell \sim \omega_{[0..\widetilde{\mathbb{M}}_j]}(j)$
$\vartheta := \emptyset$ et $X := \emptyset$
while $|X| < \ell$ **do**
  $i \sim u([1..\widetilde{\mathbb{M}}_j] \setminus \vartheta)$
  $k := \min\{l \in [1..K] \ : \ i \leq \sum_{m=1}^{k} \mathbb{M}[j][m]\}$
  $i' := \sum_{m=1}^{k} \mathbb{M}[j][m] - i + 1$
  $X := X \cup \{\texttt{itemAt}(i', j, \mathcal{D}_k)\}$ et $\vartheta := \vartheta \cup \{i\}$
od
return X

Cette section présente notre algorithme DDSAMPLING (pour Distributed Database Sampling, voir Algorithme 2) dont la force est de ne centraliser aucun item (sauf ceux qui seront tirés pour construire les motifs). Il prend en entrée une base de données distribuée et une fonction d'utilité fondée sur la taille afin de retourner un itemset X tiré proportionnellement suivant son intérêt $supp(X, \mathcal{D}) \times u(X)$. L'idée clé de notre approche consiste à centraliser la taille de chacune des transactions des différents fragments (plutôt que tous les items). Ces seules informations permettent ensuite de calculer aisément les différentes distributions de tirage et permettent de localiser les fragments utiles à la construction de l'itemset à tirer.

Matrice de pondération Nous commençons par introduire la matrice de pondération qui quantifie pour chaque transaction combien d'items sont stockés sur chaque fragment :

Définition 3 (Matrice de pondération) *La matrice de pondération $\mathbb{M}$ de la base de données distribuée $\mathcal{P} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$ est une matrice d'entiers de dimension $|\mathcal{D}| \times |\mathcal{P}|$ où $\mathbb{M}[j][k] =$ $\texttt{sizeOf}(j, \mathcal{D}_k)$ pour tout $j \in [1..|\mathcal{D}|]$ et $k \in [1..K]$*

En pratique, la matrice de pondération $\mathbb{M}$ est pré-calculée hors ligne (ligne 1) et utilisée à volonté par la suite pour tirer de nombreux échantillons au sein de l'algorithme 2. Par ailleurs, nous définissons la somme des valeurs de la ligne j, i.e. $\widetilde{\mathbb{M}}_j = \sum_{k=1}^{K} \mathbb{M}[j][k]$, qui correspond à la taille de la transaction d'identifiant j. Par exemple, le tableau 1 présente la matrice de pondération pour la base de données jouet de la figure 1, et on a $\widetilde{\mathbb{M}}_1 = 1 + 2 = |\mathcal{D}[1]|$.

j	$\mathbb{M}$				$\widetilde{\mathbb{M}}_j$	$\sum_{l=0}^{\widetilde{\mathbb{M}}_j} \omega_l^{freq}(j) =$	$\omega^{freq}(j)$	$\omega^{area}(j)$	$\omega^{\leqslant 2}(j)$
1	1	2	0	0	3	1 + 3 + 3 + 1 =	8	12	7
2	0	2	1	1	4	1 + 6 + 4 + 4 + 1 =	16	32	11
3	0	0	0	3	3	1 + 3 + 3 + 1 =	8	12	7
4	2	0	2	0	4	1 + 6 + 4 + 4 + 1 =	16	32	11
5	2	1	0	0	3	1 + 3 + 3 + 1 =	8	12	7

TAB. 1: Matrice de pondération $\mathbb{M}$ et poids pour le tirage

Calcul des poids $\omega(\mathbf{j})$ et $\omega_\ell(\mathbf{j})$ La première utilité de la matrice de pondération est de permettre un calcul aisé des poids $\omega(j)$ et $\omega_\ell(j)$. Pour rappel, ces poids correspondent respectivement à la somme des utilités des motifs de la transaction j et à la somme des utilités des motifs de taille ℓ dans la transaction j. Tout d'abord notons que la somme total $\omega(j)$ correspond à la somme de tous les poids $\omega_\ell(j)$: $\omega(j) = \sum_{\ell=0}^{\widetilde{\mathbb{M}}_j} \omega_\ell(j)$. Concentrons-nous maintenant sur le calcul de la somme des utilités des motifs de taille ℓ dans la transaction j. Intuitivement, comme tous les motifs de même taille ont la même utilité, il suffit de dénombrer le nombre de motifs de taille ℓ et de multiplier cette quantité par l'utilité $f_u(\ell)$. La propriété suivante formalise ce calcul :

Propriété 1 *Soient $\mathbb{M}$ la matrice de pondération d'une base de données distribuée $\mathcal{P} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$ et une fonction d'utilité fondée sur la taille u. Le poids des itemsets de longueur ℓ de la transaction j est défini par : $\omega_\ell(j) = \sum_{X \subseteq \mathcal{D}[j] \wedge |X| = \ell} u(X) = \binom{\widetilde{\mathbb{M}}_j}{\ell} \times f_u(\ell)$.*

L'utilisation de cette propriété est illustrée dans le tableau 1 pour trois fonctions d'utilités fondées sur la taille, u_{freq}, u_{area} et $u_{\leq 2}$. Par exemple, comme $f_{u_{freq}}(\ell) = 1$ pour toute taille ℓ,

on a $\omega^{freq}(1) = \sum_{\ell=0}^{3} \binom{3}{\ell} = 1+3+3+1 = 8 = 2^3$ (où $\binom{3}{\ell}$ est le nombre d'itemsets de taille ℓ dans une transaction de taille 3). En considérant l'aire, on a $\omega^{area}(1) = \sum_{\ell=0}^{3} \binom{3}{\ell} \times \ell = (1 \cdot 0) + (3 \cdot 1) + (3 \cdot 2) + (1 \cdot 3) = 3 + 6 + 3 = 12 = 3 \cdot 2^{3-1}$ car $f_{u_{area}}(\ell) = \ell$. Enfin, avec la contrainte de cardinalité maximale de 2, on a $\omega^{\leqslant 2}(1) = \sum_{\ell=0}^{2} \binom{3}{\ell} = 1 + 3 + 3 = 7$.

Ainsi, dans l'algorithme 2, le tirage de l'identifiant de transaction j et le tirage de la longueur ℓ de l'itemset qui sera échantillonné (ligne 3-5) reprennent le principe de l'algorithme 1. Seule la méthode de calcul des distributions diffère en s'appuyant sur la matrice $\mathbb{M}$ grâce à la propriété 1.

Construction de l'itemset Les lignes 4 à 11 détaillent le tirage de l'itemset sur l'ensemble des fragments. L'idée est de tirer sans remise une position d'item i dans la transaction d'identifiant j (ligne 8) et de rechercher le fragment k disposant de cet item (ligne 9). Il faut alors calculer la position i' de cet item au sein du fragment k (ligne 10) avant de l'interroger (ligne 11). On répète cette procédure ℓ fois (ligne 7) et pour éviter de tirer deux fois le même item, on maintient l'ensemble des positions déjà tirées ϑ. Avec l'exemple du tableau 1, si on a le tirage de la position 2 (i.e., $i = 2$) dans la transaction 1, le fragment retenu est $k = 2$ avec $i' = 2 - 1 = 1$, et l'item $\texttt{itemAt}(1, 1, \mathcal{D}_2) = B$ est ajouté à X.

4.3 Analyse théorique de DDSAMPLING

La propriété suivante démontre que DDSAMPLING retourne un échantillon de manière exacte :

Propriété 2 (Correction) *Soient $\mathcal{P} = \{D_1, ..., D_K\}$ une base de données distribuée et u une fonction d'utilité fondée sur la taille. L'algorithme 2 effectue le tirage aléatoire d'un motif X de $\mathcal{L}$ proportionnellement à $supp(X, \mathcal{D}) \times u(X)$.*

La complexité de la méthode DDSAMPLING se décompose en deux phases : celle du prétraitement correspondant à la pondération des lignes de la matrice, et celle du tirage d'un motif. Pour chaque phase, il est pertinent de déterminer la complexité en temps et en communication.

Complexité temporelle Pour le prétraitement, DDSAMPLING remplit la matrice $\mathbb{M}$ avec une complexité de $O(|\mathcal{D}| \cdot |\mathcal{P}|)$. Ensuite, il pondère les lignes de la matrice avec une complexité de $O(|\mathcal{D}| \cdot |\mathcal{I}|)$ à cause de la fonction binomiale. La complexité temporelle du prétraitement est donc $O(|\mathcal{D}| \cdot (|\mathcal{P}| + |\mathcal{I}|))$. Pour le tirage d'un motif, la première étape consiste à sélectionner une ligne de la matrice, soit une complexité de $O(log(|\mathcal{D}|))$. Deuxièmement, on tire uniformément un itemset de la transaction ayant pour identifiant le numéro de la ligne précédemment tiré avec une complexité de $O(|\mathcal{I}|)$. Finalement, nous obtenons une complexité temporelle de tirage d'un motif égale à $O(log(|\mathcal{D}|) + |\mathcal{I}|)$.

Complexité en communication Pour rappel, les coûts de communication correspondent aux requêtes `sizeOf` et `itemAt`. Dès lors, pour le prétraitement, la construction de la matrice de pondération $\mathbb{M}$ requiert $O(|\mathcal{D}_1| + \cdots + |\mathcal{D}_K|)$ échanges ce qui est souvent très inférieur au coût de la centralisation complète en $O(||\mathcal{D}_1|| + \cdots + ||\mathcal{D}_K||) = O(||\mathcal{D}||)$ échanges. Dans notre exemple jouet, le calcul de la matrice de pondération requiert 10 requêtes contre 17 pour la centralisation complète de $\mathcal{D}$. Pour le tirage d'un motif, il est nécessaire d'effectuer autant de requêtes `itemAt` que d'items contenus dans l'itemset à retourner. La complexité temporelle moyenne est donc égale à la longueur moyenne $\overline{l}$ des itemsets tirés, cette longueur étant donnée par la formule suivante : $\overline{l} = \sum_{\ell=1}^{l_{max}} \frac{\sum_{j \in [1..|\mathcal{D}|]} \omega_\ell(j)}{\sum_{j \in [1..|\mathcal{D}|]} \omega(j)} \times \ell$ où $l_{max} = max_{j \in [1..|\mathcal{D}|]} |\mathcal{D}[j]|$.

5 Expérimentations

Ces expérimentations évaluent le coût de communication de DDSAMPLING par rapport à une solution centralisée et l'impact de défaillances sur sa performance (problèmes de communication ou de pannes de sites).

Protocole expérimental Dans toutes les expérimentations réalisées, nous avons choisi comme fonction d'utilité des contraintes de taille maximale ($u_{\leq M}$ avec $M \in [1..5]$) et de taille minimale ($u_{\geq m}$ avec $m = 1$). Un tel choix permet d'éviter de tirer trop de motifs peu fréquents, en particulier lorsque les jeux de données considérés contiennent de longues transactions. Les expériences ont été conduites avec 5 bases de données de l'UCI (`archive.ics.uci.edu/ml`). Pour obtenir des bases de données distribuées [1], nous avons fragmenté uniformément chaque jeu de données en $K = 10$ fragments. Pour générer un partitionnement horizontal, chaque transaction est placée aléatoirement dans un fragment donné avec la même probabilité $1/K$. Pour un partitionnement hybride, tous les items d'une transaction sont placés indépendamment et aléatoirement (avec la même probabilité $1/K$) dans un fragment donné. Enfin, pour générer un partitionnement vertical, chaque item est placé aléatoirement sur un site donné (avec la même probabilité $1/K$). La table 2 récapitule les caractéristiques des jeux données utilisés dans les trois premières colonnes (nombre d'items, de transactions et taille globale). Toutes les expérimentations sont réalisées sur un PC de 2.71 GHz 2 Core CPU avec une RAM de 12 GB.

Bases centralisées				Nb appels de `sizeOf` Bases distribuées			Nb appels de `itemAt` Bases distribuées			T_{max}
benchmarks	$\|\mathcal{I}\|$	$\|\mathcal{D}\|$	$\|\|\mathcal{D}\|\|$	Horizontal	Hybride	Vertical	M=1	M=3	M=5	si vertical
`Chess`	75	3 196	118 252	3 196	31 312	31 427	1.0	2.91	4.83	86 825
`Connect`	129	67 557	2 904 951	67 557	668 296	668 547	1.0	2.92	4.86	2 236 404
`Iris`	15	150	750	150	614	618	1.0	2.19	2.58	132
`Mushroom`	119	8 124	186 852	8 124	74 036	74 180	1.0	2.85	4.7	112 816
`Waveform`	67	5 000	110 000	5 000	45 081	45 324	1.0	2.84	4.68	64 676

TAB. 2: Caractéristiques des benchmarks et coûts de communication

Coûts de communication En phase de prétraitement, le coût de communication correspond au nombre d'appels à la requête `sizeOf` pour construire la matrice de pondération. Comme le montre la table 2, il est naturellement plus élevé dans le cas de partitionnements hybride et vertical, les items d'une transaction n'étant pas nécessairement sur le même fragment. En phase de tirage, le coût de communication correspond au nombre d'appels à la requête `itemAt`. La table 2 montre le nombre moyen d'appels pour le tirage d'un motif. Notons qu'il est indépendant du type de partitionnement, mais dépend de la contrainte de cardinalité maximale considérée ($M \in \{1, 3, 5\}$), cette contrainte jouant sur la taille moyenne des motifs tirés. Pour finir, il est intéressant de comparer le coût de communication d'une solution distribuée par rapport à une solution centralisée en évaluant dans le pire des cas (i.e., partitionnement vertical et $M = 5$) le nombre de motifs T_{max} (dernière colonne de la table 2) qu'il est possible de tirer avant d'atteindre un coût de communication équivalent à la centralisation. Excepté pour la base `Iris` de toute petite taille, on note ainsi qu'une approche décentralisée permet de tirer quelques milliers de motifs avec un coût de communication bien inférieur à celui d'une approche centralisée.

1. Jeux de données et code source disponibles à `github.com/DDSampling/DDSampling`

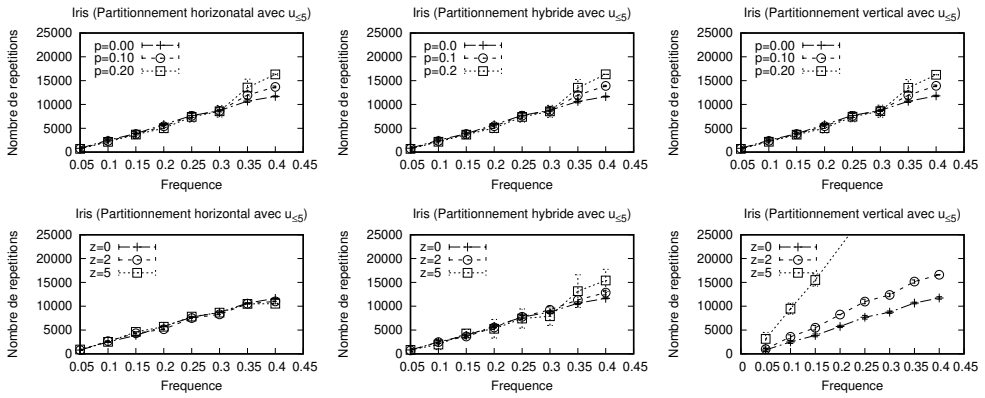

FIG. 2: Qualité de DDSAMPLING face aux défaillances de communication ou aux pannes

Altération de l'exactitude du tirage Avec les bases de données distribuées, deux des principaux problèmes sont les erreurs de communication réseau ou l'inaccessibilité d'un site. Chacun de ces phénomènes induit un biais dans les tirages réalisés. En effet, lors du tirage d'un motif X, nous le rejetons systématiquement en cas d'échec d'une des requêtes `itemAt` nécessaires à sa construction. Pour mesurer le biais résultant d'une telle stratégie, nous avons utilisé le jeu de données `Iris` pour tirer un million de motifs de taille maximale $M = 5$. Ensuite, nous avons évalué si les motifs de l'échantillon ont effectivement été tirés avec une probabilité proportionnelle à leur mesure d'intérêt. Pour ce faire, nous comptons pour chaque tranche de fréquence $[\theta, \theta + 0.05[$ le nombre moyen de répétitions (ainsi que son écart type) avec lequel un motif apparait dans l'échantillon alors que sa fréquence réelle est dans la tranche considérée. Si le tirage est exact, les motifs sont tirés proportionnellement à leur mesure d'intérêt et la courbe doit être une droite.

En suivante ce protocole, nous avons évalué la qualité des échantillons construits en faisant varier la probabilité $p \in \{0.00, 0.10, 0.20\}$ qu'une communication avec un site échoue, ou qu'un nombre $z \in \{0, 2, 5\}$ de sites soient en panne (les sites en panne étant tirés aléatoirement). Les résultats obtenus sont représentés à la figure 2. De manière générale, les courbes obtenues montrent que les défaillances n'altèrent pas significativement l'exactitude du tirage si leur niveau reste modéré ($p \leq 10\%$ et au plus $z = 2$ sites sur $K = 10$ sont en panne), et ceci quel que soit le type de partitionnement. Plus précisément, dans le cas de problèmes de communication, les motifs les plus fréquents sont les plus tirés car ils sont généralement les plus courts et ont ainsi moins de chance d'être rejetés. Ils seront donc sur-représentés dans l'échantillon construit. Dans le cas de sites en panne permanente, on constate que la probabilité de tirage d'un motif reste proportionnelle à sa mesure d'intérêt. Pour le cas vertical, tous les motifs tirés le restent de manière exacte (leurs items étant toujours accessibles au cours du temps). Plus le nombre de sites en panne est important, moins il y a de motifs disponibles. Du coup, les motifs restants sont plus fréquemment tirés augmentant la pente des courbes.

Taux de motifs rejetés Nous évaluons maintenant le taux de rejet moyen pour tirer 10 000 motifs en variant p et z. L'expérience est répétée 100 fois en faisant varier aléatoirement le

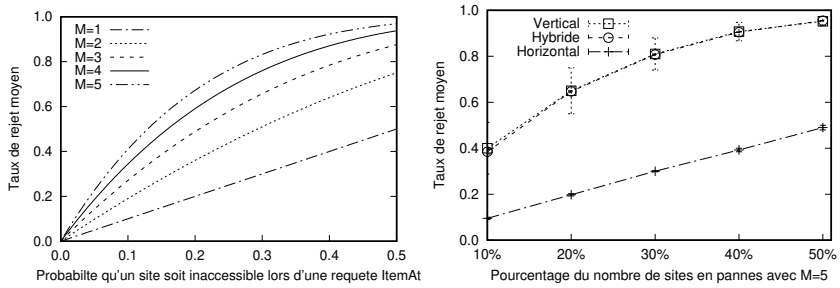

FIG. 3: Nombre moyen de rejets lors d'une défaillance

partitionnement des bases et les sites en panne. La figure 3 montre les taux de rejets moyens et leurs écarts-types en calculant la moyenne des taux de rejets obtenus pour les 5 bases : `Chess`, `Connect`, `Iris`, `Mushroom` et `Waveform`. En effet, ces taux sont indépendants d'un jeu de données particulier, les partitionnements générés étant aléatoires et uniformes. Les taux de rejets étant indépendants d'un type particulier de partionnement, la partie gauche de la figure 3 représente l'évolution du taux de rejet moyen en fonction de p pour $M \in \{1, 2, 3, 4, 5\}$. Pour une valeur de p fixe, on constate que ce taux de rejet augmente avec M. Néanmoins, il reste inférieur à 50% si p est inférieur à 0.1 ce qui constitue déjà un niveau élevé de défaillance. La partie droite de la figure 3 représente les taux de rejets en cas de panne de sites. Tout d'abord, ils sont plus faibles dans le cas d'un partitionnement horizontal où le taux de rejet est égal à z/K. En moyenne, ils sont plus élevés dans le cas de partitionnements hybride ou vertical. Enfin, l'écart-type des taux de rejets moyen est plus élevé dans le cas d'un partitionnement vertical. En effet, avec un partitionnement vertical, les taux de rejets seront plus ou moins élevés si les items des motifs les plus fréquents sont ou pas sur les sites en panne. Pour finir, on note que le taux de rejet moyen reste inférieur à 50% si moins de 10% des sites sont en panne. Comme pour le cas de défaillances de communication, ce taux de rejet reste acceptable du point de vue du temps de calcul nécessaire pour construire un échantillon.

6 Conclusion

Cet article propose le premier algorithme d'échantillonnage de motifs ensemblistes depuis une base de données distribuée. Il permet de considérer différentes mesures d'intérêt et surtout, des partitionnements hybrides ou verticaux. Grâce à la seule centralisation des tailles, les coûts de communications de l'approche sont raisonnables car les échanges d'items sont réalisés uniquement lorsque l'utilisateur demande des motifs. De plus, l'étude expérimentale souligne la robustesse de l'approche. Dans les travaux futurs, nous envisageons de remplacer le tirage exact des transactions par une méthode stochastique afin de ne plus avoir à centraliser les tailles de toutes les transactions de chacun des fragments. Il serait également intéressant de proposer un mécanisme de correction des poids afin de contre-balancer la panne d'un site.

Remerciements. Lamine Diop est partiellement financé par le CEA-MITIC, Centre d'Excellence Africain en Mathématiques, Informatique et TIC.

Références

Al Hasan, M. et M. J. Zaki (2009). Output space sampling for graph patterns. *Proc. of the VLDB Endowment 2*(1), 730–741.

Boley, M., C. Lucchese, D. Paurat, et T. Gärtner (2011). Direct local pattern sampling by efficient two-step random procedures. In *Proc. of the 17th ACM SIGKDD*, pp. 582–590.

Cheung, D. W., V. T. Ng, A. W. Fu, et Y. Fu (1996). Efficient mining of association rules in distributed databases. *IEEE transactions on Knowledge and Data Engineering 8*(6), 911–922.

Dietterich, T. G. et R. S. Michalski (1983). A comparative review of selected methods for learning from examples. In *Machine Learning*, pp. 41–81. Springer.

Diop, L., C. T. Diop, A. Giacometti, D. L. Haoyuan, et A. Soulet (2018). Sequential pattern sampling with norm constraints. In *IEEE International Conference on Data Mining (ICDM)*.

Giacometti, A. et A. Soulet (2016). Anytime algorithm for frequent pattern outlier detection. *International Journal of Data Science and Analytics 2*(3-4), 119–130.

Giacometti, A. et A. Soulet (2017). Interactive pattern sampling for characterizing unlabeled data. In *Proc. of IDA 2017*, pp. 99–111. Springer.

Giacometti, A. et A. Soulet (2018). Dense neighborhood pattern sampling in numerical data. In *Proc. of SDM 2018*, pp. 756–764.

Jin, R. et G. Agrawal (2006). Systematic approach for optimizing complex mining tasks on multiple databases. In *22nd International Conference on Data Engineering (ICDE'06)*, pp. 17–17.

Kum, H.-C., J. H. Chang, et W. Wang (2006). Sequential pattern mining in multi-databases via multiple alignment. *Data Mining and Knowledge Discovery 12*(2-3), 151–180.

Moens, S. et M. Boley (2014). Instant exceptional model mining using weighted controlled pattern sampling. In *Proc. of IDA 2014*, pp. 203–214. Springer.

Otey, M. E., C. Wang, S. Parthasarathy, A. Veloso, et W. Meira (2003). Mining frequent itemsets in distributed and dynamic databases. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pp. 617–620. IEEE.

Özsu, M. T. et P. Valduriez (2011). *Principles of distributed database systems*. Springer Science & Business Media.

Zhang, S. et M. J. Zaki (2006). Mining multiple data sources : local pattern analysis. *Data Mining and Knowledge Discovery 12*(2-3), 121–125.

Zhu, X., B. Li, X. Wu, D. He, et C. Zhang (2011). Clap : Collaborative pattern mining for distributed information systems. *Decision support systems 52*(1), 40–51.

Zhu, X. et X. Wu (2007). Discovering relational patterns across multiple databases. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pp. 726–735. IEEE.

Summary

Only few pattern mining methods are dedicated to distributed databases. In fact, the centralization of data is often less expensive than the communication of all mined patterns. To circumvent this difficulty, this paper follows a parsimonious approach by sampling patterns. We propose the algorithm DDSAMPLING that draws a pattern from a distributed database proportionally to its interest. We demonstrate its accuracy and analyze its complexity. Experiments show on several datasets its robustness against the failures of a site or the network.