Recherche de motifs séquentiels dans les data streams

Chedy Raissi

LGI2P/EMA - LIRMM

20 mai 2005

Plan

Contexte

- Un nouveau problème : des données sous la forme d'un flot continu et rapide
- De nombreux domaines d'applications :
  - mesure du trafic de réseau
  - analyse de tendances
  - analyse de comportements sur le Web
- Approches classiques non-adaptées au flot continu de données

Un bref rappel

Les Motifs Séquentiels (1/2)

Definitions

- **Itemset** : ensemble d'items
- **Transaction**
  un n-uplet ($client_{id}$, *itemset*, *date*)
  - $client_{id}$: identifiant unique du client
  - *itemset* : ensemble d'items achetés par le client $client_{id}$
  - *date* : date de l'achat
- **Séquence** : une liste ordonnée d'itemsets

Les Motifs Séquentiels (2/2)

Définition

- **Support d'une séquence** : nombre d'occurrences de la séquence de données dans la base DB

$$S_1 = \langle a_1, a_2, \ldots, a_n \rangle \preceq S_2 = \langle b_1, b_2, \ldots, b_m \rangle \\ \Leftrightarrow \exists i_1 < \ldots < i_n \mid a_1 \subseteq b_{i_1}, \ldots, a_n \subseteq b_{i_n}$$

Problématique

Rechercher toutes les séquences S fréquentes, i.e. vérifiant $supp(S) \geq \sigma$ dans DB

Exemple :

$\langle (20)\ (40) \rangle \preceq \langle (10)\ (\mathbf{20}\ 30)\ (\mathbf{40})\ (20) \rangle$

$\langle (20)\ (30) \rangle \not\preceq \langle (10)(20\ 30)(40)(20) \rangle$

Prise en compte des data streams

Quelques définitions

- Batch B_i^j : ensemble de séquences de données, $B_i^j = [S_1, S_2, \ldots, S_k]$
- Data Stream $DS = B_0^0, B_1^1, \ldots, B_n^n \ldots$ ensemble infini de batches
- Longueur du DS $L = |B_0^0| + |B_1^1| + \ldots + |B_n^n|$

Problèmatique

- Rechercher toutes les séquences vérifiant :

$$\sum_{t=0}^{L} support_t(S) \geq \sigma.L$$

Illustration

B_0^0	S_a	(1) (2) (3) (4) (5)
	S_b	(8) (9)
B_1^1	S_c	(1) (2)
B_2^2	S_d	(1) (2) (3)
	S_e	(1) (2) (8) (9)
	S_f	(2) (1)

Fig.: Les ensembles de batches B_0^0, B_1^1 et B_2^2

Avec $\sigma = 50\%$, nous avons les séquences fréquentes suivantes

- $[t_0 - t_0]$: $\langle(1)(2)(3)(4)(5)\rangle$ et $\langle(8)(9)\rangle$
- $[t_0 - t_1]$: $\langle(1)(2)\rangle$
- $[t_0 - t_2]$: $\langle(1)(2)\rangle$, $\langle(1)\rangle$ et $\langle(2)\rangle$

Travaux Antérieurs

Les approches traditionnelles 1/2

- De nombreuses approches : GSP, SPADE, PSP...

Algorithme Générer/Elaguer

1. Recherche des items fréquents (1-séquences fréquentes)
2. Génération des candidats de taille 2 en combinant les 1-séquences fréquentes
3. Parcours sur la base (2-séquences fréquentes)
4. Génération des candidats de taille n en combinant (jointure particulière) les (n-1) séquences fréquentes
5. Parcours sur la base (n-séquences fréquentes)
6. L'algorithme se termine quand plus aucun candidat n'est généré

Les approches traditionnelles 2/2

- D'autres approches : bases stockées en mémoire, PrefixSpan
- Génération : un ensemble d'opérations de jointure (opération bloquante) [Babcock et al, 2002]

Remarque

Les éléments de *DB* sont examinés plus d'une fois

Les approches incrémentales

- Prise en compte de l'évolution des *DB* au cours du temps
- ISE : utilisation d'informations minimales pour optimiser le nombre de candidats à tester
- ISM (extension de SPADE) maintient une bordure négative

Remarque

Plusieurs passes sur la base

Les nouvelles approches 1/2

- Les approches existantes : associations **inter-transactions**

1. Manku-Motwani (*VLDB'02*) : une passe, concept de la ϵ-**synopsis function**, approximation des résultats
2. Li et al (*1st Workshop on Data Stream'2004*) : utilisation d'un arbre préfixé
3. Chi et al (*ICDM'04*) : extraction d'itemsets fermés
4. Giannella et al. (*Next Generation Data Mining, 2003*) : algorithme inspiré des FP-trees. Définition des Tilted-Time Windows logarithmiques

Les nouvelles approches 2/2

Les requêtes impossibles

1. Quels sont les motifs séquentiels fréquents sur la période allant de t_i à t_j ?
2. Quelles sont les périodes où la séquence S est fréquente ?
3. Est-ce que le support de la fréquence S change entre le temps t_a et t_b ?

Tilted-Time Window Naturelle

- Pour répondre à ce type de requêtes il faut définir et maintenir un historique
- Une structure représentative du temps appropriée : les Tilted-Time Windows [Chen et al,2002]
- Un concept de granularité temporelle : on s'intéresse avec plus de précisions aux faits récents qu'aux anciens faits

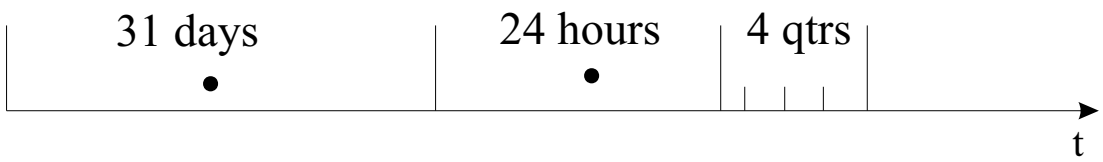


Fig.: Tilted-Time Window représentant un mois

Tilted-Time Window Logarithmique

- Besoin d'optimiser l'espace : concept des fenêtres logarithmiques [Gianella et al,2003]

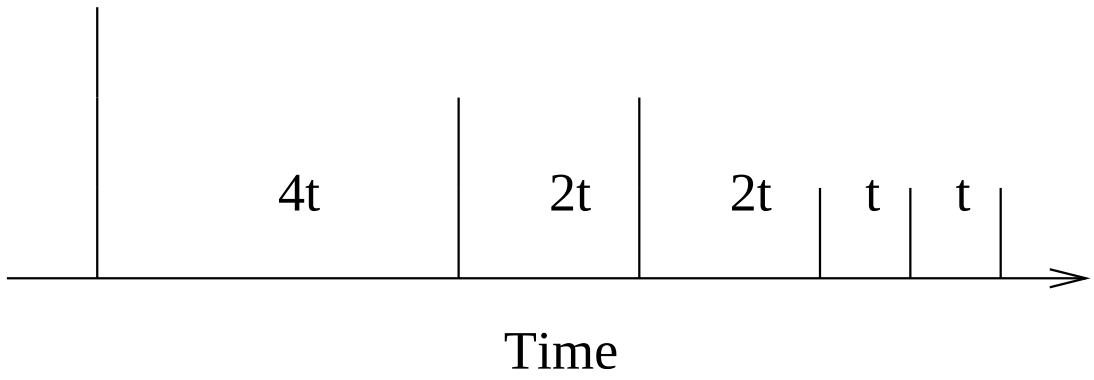


Fig.: Tilted-Time Window logarithmique

Speed : Une nouvelle approche

Un exemple

B_0^0	S_a	(1) (2) (3) (4) (5)
	S_b	(8) (9)
B_1^1	S_c	(1) (2)
B_2^2	S_d	(1) (2) (3)
	S_e	(1) (2) (8) (9)
	S_f	(2) (1)

Traitement de la première séquence

- $S_a = \langle(1)(2)(3)(4)(5)\rangle$
- Mise à jour de la structure *ITEMS* et de l'arbre de séquences *TreeVal*

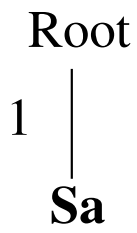
Un exemple

Items	Tilted	*Valuations*	$Root_{Val}$
1	$[t_0,1]$	1	S_a
2	$[t_0,1]$	1	S_a
3	$[t_0,1]$	1	S_a
4	$[t_0,1]$	1	S_a
5	$[t_0,1]$	1	S_a

Fig.: La structure *ITEMS*

- *Tilted* indique la tilted-time window associée à l'item
- *Valuation* indique la valuation de l'item
- $Root_{Val}$ est un pointeur sur la séquence de l'arbre

Un exemple



(Premier Batch – Sa)

Fig.: La structure *TreeVal* après le premier traitement

Quelques détails...

- Chaque séquence est associée à une valuation : recherche et élagage plus rapide
- Les séquences sont ordonnées en fonction de leurs inclusions
- On ne stocke que les séquences maximales

Un exemple

Traitement de la seconde séquence

- $S_b = \langle(8)(9)\rangle$

Items	Tilted	*Valuations*	$Root_{Val}$
1	$[t_0,1]$	1	S_a
2	$[t_0,1]$	1	S_a
3	$[t_0,1]$	1	S_a
4	$[t_0,1]$	1	S_a
5	$[t_0,1]$	1	S_a

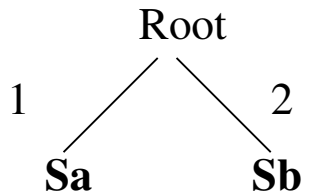
Un exemple

Traitement de la seconde séquence

- $S_b = \langle(8)(9)\rangle$
- Les items 8 et 9 n'existent pas dans *ITEMS* : création d'une nouvelle valuation

Items	Tilted	*Valuations*	$Root_{Val}$
1	$[t_0,1]$	1	S_a
2	$[t_0,1]$	1	S_a
3	$[t_0,1]$	1	S_a
4	$[t_0,1]$	1	S_a
5	$[t_0,1]$	1	S_a
8	$[t_0,1]$	2	S_b
9	$[t_0,1]$	2	S_b

Un exemple



(Premier Batch – Sa, Sb)

Fig.: La structure *TreeVal* après la seconde séquence

Traitement de la seconde séquence (suite)

- insertion de la séquence dans *TreeVal* avec une nouvelle valuation

Un exemple

Traitement de la troisième séquence

- $S_c = \langle(1)(2)\rangle$

Items	Tilted	*Valuations*	$Root_{Val}$
1	$[t_0,1]$ $[t_1,1]$	1	S_a
2	$[t_0,1]$ $[t_1,1]$	1	S_a
...	...	...	...

Un exemple

Traitement de la troisième séquence

- $S_c = \langle(1)(2)\rangle$
- Les items 1 et 2 existent déjà dans la structure *ITEMS*

Items	Tilted	*Valuations*	$Root_{Val}$
1	$[t_0,1]$ $[t_1,1]$	1	S_a
2	$[t_0,1]$ $[t_1,1]$	1	S_a
...	...	...	...

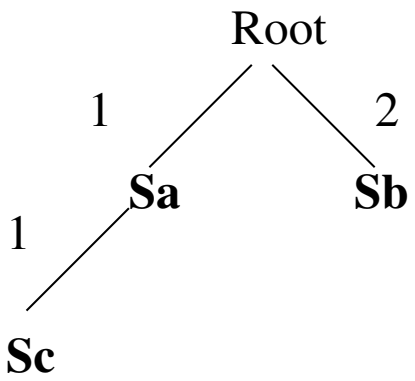
Un exemple

Traitement de la troisième séquence

- $S_c = \langle(1)(2)\rangle$
- Les items 1 et 2 existent déjà dans la structure *ITEMS*
- Leur valuation est de 1, S_c est-elle incluse dans S_a ?

Items	Tilted	*Valuations*	$Root_{Val}$
1	$[t_0,1]$ $[t_1,1]$	1	S_a
2	$[t_0,1]$ $[t_1,1]$	1	S_a
...	...	...	...

Un exemple



(Second Batch – Sc)

Fig.: La structure *TreeVal* après la troisième séquence

Traitement de la troisième séquence (suite)

- L'unique sous-séquence commune à S_a et S_c c'est S_c !
- Il suffit donc de mettre à jour *ITEMS* et *TreeVal*

Un exemple

Traitement de la quatrième séquence

- $S_d = \langle(1)(2)(3)\rangle$
- Les items 1, 2 et 3 existent déjà dans la structure *ITEMS*
- Leur valuation est de 1, la plus longue sous-séquence commune à S_a et S_d est $\langle(1)(2)(3)\rangle$
- *ITEMS* et *TreeVal* sont mis à jour

Un exemple

Traitement de la cinquième séquence

- $S_e = \langle(1)(2)(8)(9)\rangle$
- Tous les items existent déjà dans *ITEMS*
- Deux valuations différentes : 1 et 2

Items	Tilted	*Valuations*	$Root_{Val}$
1	$[t_0,1]$	1	S_a
2	$[t_0,1]$	1	S_a
3	$[t_0,1]$	1	S_a
4	$[t_0,1]$	1	S_a
5	$[t_0,1]$	1	S_a
8	$[t_0,1]$	2	S_b
9	$[t_0,1]$	2	S_b

Un exemple

Traitement de la cinquième séquence

- $S_e = \langle(1)(2)(8)(9)\rangle$
- Tous les items existent déjà dans *ITEMS*
- Deux valuations différentes : 1 et 2

Items	Tilted	*Valuations*	$Root_{Val}$
1	$[t_0,1]$	1	S_a
2	$[t_0,1]$	1	S_a
3	$[t_0,1]$	1	S_a
4	$[t_0,1]$	1	S_a
5	$[t_0,1]$	1	S_a
8	$[t_0,1]$	2	S_b
9	$[t_0,1]$	2	S_b

Un exemple

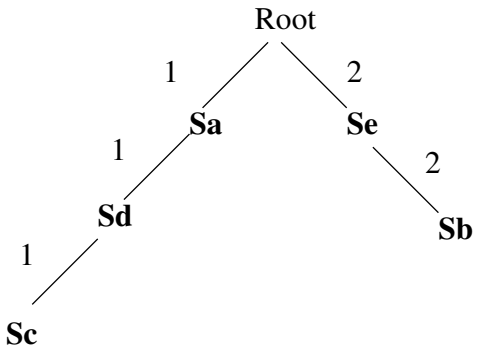
Traitement de la cinquième séquence (suite)

- La plus longue sous-séquence maximale pour la valuation 1 est : $\langle(1)(2)\rangle$
- La plus longue sous-séquence maximale pour la valuation 2 est : $\langle(8)(9)\rangle$
- Insertion de S_e comme racine de la valuation 2
- Insertion de $\langle(1)(2)\rangle$ dans la branche de l'arbre de valuation 1

Un exemple

Items	Tilted	*Valuations*	$Root_{Val}$
1	$[t_0,1]$ $[t_1,1]$ $[t_2,1]$ $[t_3,1]$	1 2	S_a S_e
2	$[t_0,1]$ $[t_1,1]$ $[t_2,1]$	1 2	S_a S_e
...	...	..	...
8	$[t_0,1]$ $[t_3,1]$	2	S_e
9	$[t_0,1]$ $[t_3,1]$	2	S_e

Un exemple



(Troisieme Batch – Se)

Fig.: La structure *TreeVal* après la cinquième séquence

Un exemple

Traitement de la sixième séquence

- $S_f = \langle(2)(1)\rangle$
- Tous les items existent déjà dans *ITEMS*
- Il existe deux valuations différentes : 1 et 2
- La plus longue sous-séquence maximale pour la valuation 1 est : $\emptyset$
- La plus longue sous-séquence maximale pour la valuation 2 est : $\emptyset$
- C'est une nouvelle séquence non incluse dans aucune autre : nouvelle valuation

Un exemple

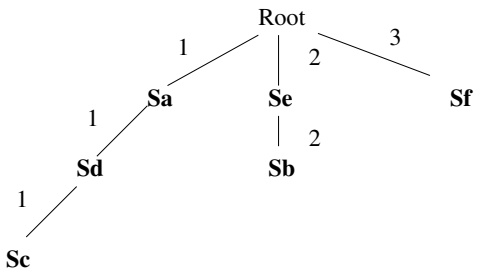


Fig.: La structure *TreeVal* après la dernière séquence

Un exemple

Elagage des séquences non-fréquentes

- Maintenir tous les data streams en mémoire centrale nécessite trop d'espace
- Elagage répété à chaque fin de traitement des batches
- 2 types d'élagages :
  1. $\text{support}_t(S) < \epsilon\ |B_i^i|$
  2. $\exists l, \forall i, l \leq i \leq n, support_{t_i}(S) < \sigma.|B_i^i|$
     et
     $\forall l',\ l \leq m \leq l' \leq n,\ \sum_{i=l}^{l'} \text{support}_{t_i}(S) < \varepsilon.\sum_{i=l}^{l'} |B_i^i|$

Approximations des résultats

- Conséquence de l'élagage : une approximation de la fréquence exacte sur L
- Avec $\varepsilon \ll \sigma$, cette approximation est assurée d'être plus petite que la fréquence actuelle : $F_S(L) - \epsilon|L| \leq \tilde{F}_S(L) \leq F_S(L)$ [Giannella et al, 2003].

Une nouvelle représentation des itemsets 1/2

- Une transformation pendant le pré-traitement afin de diminuer le nombre de tests d'inclusions
- Une correspondance simple : à chaque item on associe un nombre premier
- Pour chaque itemset de la séquence : un nombre dont la factorisation en facteurs premiers est unique.

Une nouvelle représentation des itemsets 2/2

- Exemple :

a	b	c
2	3	5

Une nouvelle représentation des itemsets 2/2

- Exemple :

a	b	c
2	3	5

- Sequence avant traitement $S_n = \langle (abc), (bc), (abc), (ac) \rangle$

Une nouvelle représentation des itemsets 2/2

- Exemple :

a	b	c
2	3	5

- Sequence avant traitement $S_n = \langle (abc), (bc), (abc), (ac) \rangle$
- Sequence après traitement $S_p = \langle (30), (15), (30), (10) \rangle$

Une nouvelle représentation des itemsets 2/2

- Exemple :

a	b	c
2	3	5

- Sequence avant traitement $S_n = \langle (abc), (bc), (abc), (ac) \rangle$
- Sequence après traitement $S_p = \langle (30), (15), (30), (10) \rangle$

- Sequence avant traitement $P_n = \langle (bc), (ac) \rangle$

Une nouvelle représentation des itemsets 2/2

- Exemple :

a	b	c
2	3	5

- Sequence avant traitement $S_n = \langle (abc), (bc), (abc), (ac) \rangle$
- Sequence après traitement $S_p = \langle (30), (15), (30), (10) \rangle$

- Sequence avant traitement $P_n = \langle (bc), (ac) \rangle$
- Sequence après traitement $P_p = \langle (15), (10) \rangle$

Une nouvelle représentation des itemsets 2/2

- Exemple :

a	b	c
2	3	5

- Sequence avant traitement $S_n = \langle (abc), (bc), (abc), (ac) \rangle$
- Sequence après traitement $S_p = \langle (30), (15), (30), (10) \rangle$

- Sequence avant traitement $P_n = \langle (bc), (ac) \rangle$
- Sequence après traitement $P_p = \langle (15), (10) \rangle$
- Est-ce que $P \preceq S$?

Une nouvelle représentation des itemsets 2/2

- Exemple :

a	b	c
2	3	5

- Sequence avant traitement $S_n = \langle (abc), (bc), (abc), (ac) \rangle$
- Sequence après traitement $S_p = \langle (30), (15), (30), (10) \rangle$

- Sequence avant traitement $P_n = \langle (bc), (ac) \rangle$
- Sequence après traitement $P_p = \langle (15), (10) \rangle$
- Est-ce que $P \preceq S$?
- Avec un test classique : 8 étapes pour le calculer

Une nouvelle représentation des itemsets 2/2

- Exemple :

a	b	c
2	3	5

- Sequence avant traitement $S_n = \langle (abc), (bc), (abc), (ac) \rangle$
- Sequence après traitement $S_p = \langle (30), (15), (30), (10) \rangle$

- Sequence avant traitement $P_n = \langle (bc), (ac) \rangle$
- Sequence après traitement $P_p = \langle (15), (10) \rangle$
- Est-ce que $P \preceq S$?
- Avec un test classique : 8 étapes pour le calculer
- Avec un test modulo : 3 étapes

Les séquences maximales communes

- Dans SPEED on cherche à extraire les motifs séquentiels **maximaux**
- On peut se ramener au problème de recherche des plus longues sous-séquences communes [Hirschberg D., 1975][Eppstein et al,1990]
- L'approche naïve par programmation dynamique (n et m longueur des séquences) : $O(nm)$
- Notre proposition : une variante de l'algorithme NKY [Nakatsu et al,1978] : $O(n(m-r))$

Exemple de calcul

- Soit $S_a = \langle (a), (c), (b), (e), (c), (d) \rangle$
- Soit $S_b = \langle (c), (b), (c), (e), (d) \rangle$
- Quels sont leurs plus longues sous-sequences communes ?

Exemple de calcul

- Soit $S_a = \langle (a), (c), (b), (e), (c), (d) \rangle$
- Soit $S_b = \langle (c), (b), (c), (e), (d) \rangle$
- Quels sont leurs plus longues sous-sequences communes ?

	$\emptyset$	a	c	b	e	c	d	$\emptyset$
c								
b								
c								
e								
d								

Exemple de calcul

- Soit $S_a = \langle (a), (c), (b), (e), (c), (d) \rangle$
- Soit $S_b = \langle (c), (b), (c), (e), (d) \rangle$
- Quels sont leurs plus longues sous-sequences communes ?

	$\emptyset$	a	c	b	e	c	d	$\emptyset$
c							5	0
b								
c								
e								
d								

Exemple de calcul

- Soit $S_a = \langle (a), (c), (b), (e), (c), (d) \rangle$
- Soit $S_b = \langle (c), (b), (c), (e), (d) \rangle$
- Quels sont leurs plus longues sous-sequences communes ?

	$\emptyset$	a	c	b	e	c	d	$\emptyset$
c							5	0
b						3	0	
c								
e								
d								

Exemple de calcul

- Soit $S_a = \langle (a), (c), (b), (e), (c), (d) \rangle$
- Soit $S_b = \langle (c), (b), (c), (e), (d) \rangle$
- Quels sont leurs plus longues sous-sequences communes ?

	$\emptyset$	a	c	b	e	c	d	$\emptyset$
c							5	0
b						3	0	
c					0	0		
e								
d								

Exemple de calcul

- Soit $S_a = \langle (a), (c), (b), (e), (c), (d) \rangle$
- Soit $S_b = \langle (c), (b), (c), (e), (d) \rangle$
- Quels sont leurs plus longues sous-sequences communes ?

	$\emptyset$	a	c	b	e	c	d	$\emptyset$
c						5	5	0
b						3	0	
c					0	0		
e								
d								

Exemple de calcul

- Soit $S_a = \langle (a), (c), (b), (e), (c), (d) \rangle$
- Soit $S_b = \langle (c), (b), (c), (e), (d) \rangle$
- Quels sont leurs plus longues sous-sequences communes ?

	$\emptyset$	a	c	b	e	c	d	$\emptyset$
c						5	5	0
b					4	3	0	
c					0	0		
e								
d								

Exemple de calcul

- Soit $S_a = \langle (a), (c), (b), (e), (c), (d) \rangle$
- Soit $S_b = \langle (c), (b), (c), (e), (d) \rangle$
- Quels sont leurs plus longues sous-sequences communes ?

	$\emptyset$	a	c	b	e	c	d	$\emptyset$
c						5	5	0
b					4	3	0	
c				2	0	0		
e				0				
d								

Exemple de calcul

- Soit $S_a = \langle (a), (c), (b), (e), (c), (d) \rangle$
- Soit $S_b = \langle (c), (b), (c), (e), (d) \rangle$
- Quels sont leurs plus longues sous-sequences communes ?

	$\emptyset$	a	c	b	e	c	d	$\emptyset$
c						5	5	0
b					4	3	0	
c				2	0	0		
e			1	0				
d			0					

Exemple de calcul

- Soit $S_a = \langle (a), (c), (b), (e), (c), (d) \rangle$
- Soit $S_b = \langle (c), (b), (c), (e), (d) \rangle$
- Quels sont leurs plus longues sous-sequences communes ?

	$\emptyset$	a	c	b	e	c	d	$\emptyset$
c						5	5	0
b					4	3	0	
c				2	0	0		
e			1	0				
d		0	0					

Exemple de calcul

- Soit $S_a = \langle (a), (c), (b), (e), (c), (d) \rangle$
- Soit $S_b = \langle (c), (b), (c), (e), (d) \rangle$
- Quels sont leurs plus longues sous-sequences communes ?

		a	c	b	e	c	d	
c						5	5	0
b					4	3	0	
c				2	0	0		
e			1	0				
d		0	0					

Fig.: Le parcours dans la matrice

Exemple de calcul

- Soit $S_a = \langle (a), (c), (b), (e), (c), (d) \rangle$
- Soit $S_b = \langle (c), (b), (c), (e), (d) \rangle$
- Quels sont leurs plus longues sous-sequences communes ?

		a	c	b	e	c	d	
c						5	5	0
b					4	3	0	
c				2	0	0		
e			1	0				
d		0	0					

Fig.: Le parcours dans la matrice

- Première sous-séquence : $\langle (c), (b), (e), (d) \rangle$
- Deuxième sous-séquence : $\langle (c), (b), (c), (d) \rangle$

- Les buts :
  1. Le temps de calcul de notre algorithme ne doit pas dépasser la largeur des batches
  2. Majorer l'espace mémoire utilisé
- 2 types de jeu de séquences : longues et courtes

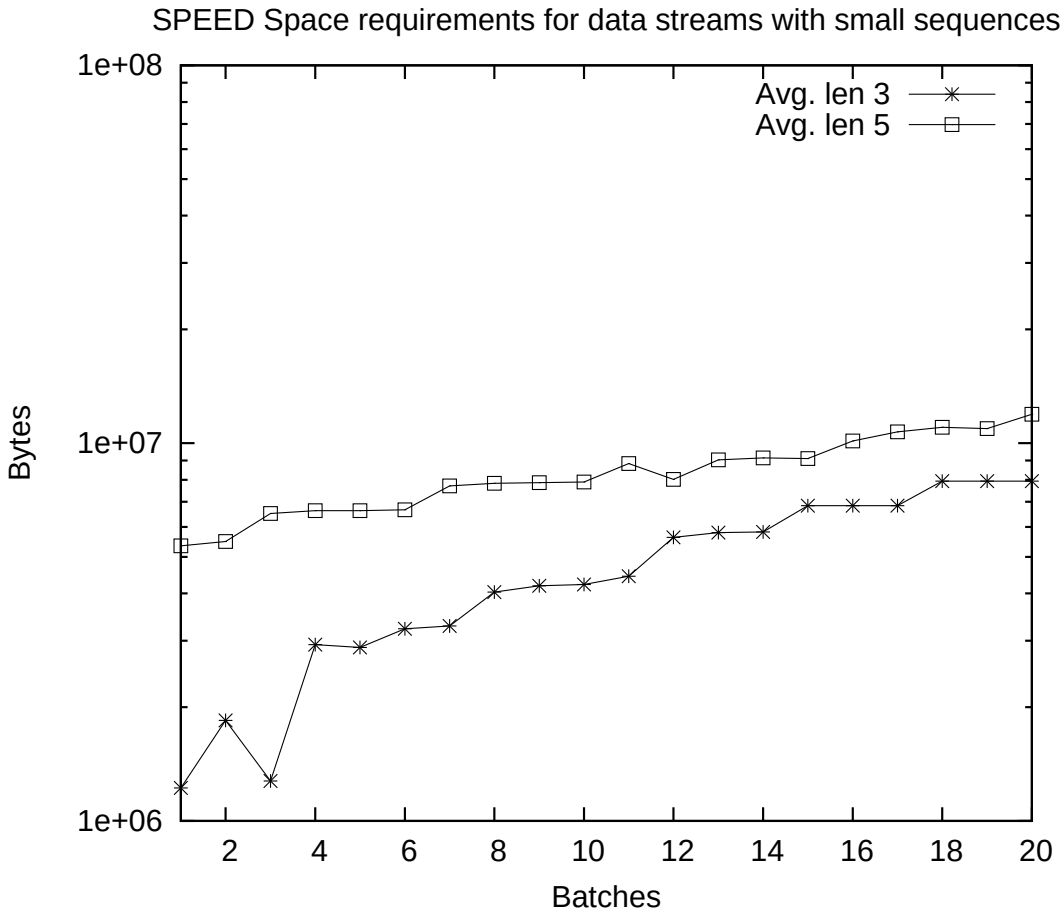


Fig.: Memoire utilisée pour les petites séquences $\sigma = 0.1$ *et* $\epsilon = 0.075$

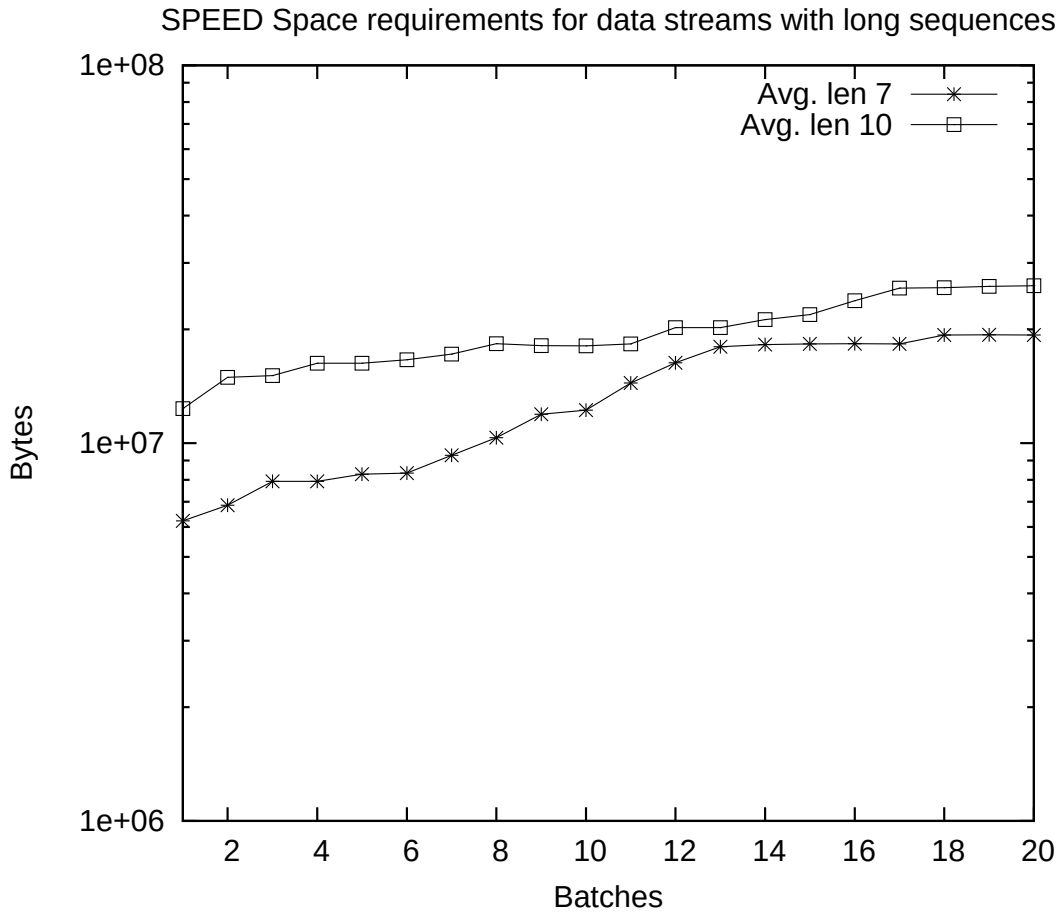


Fig.: Memoire utilisée pour les longues séquences $\sigma = 0.1$ *et* $\epsilon = 0.075$

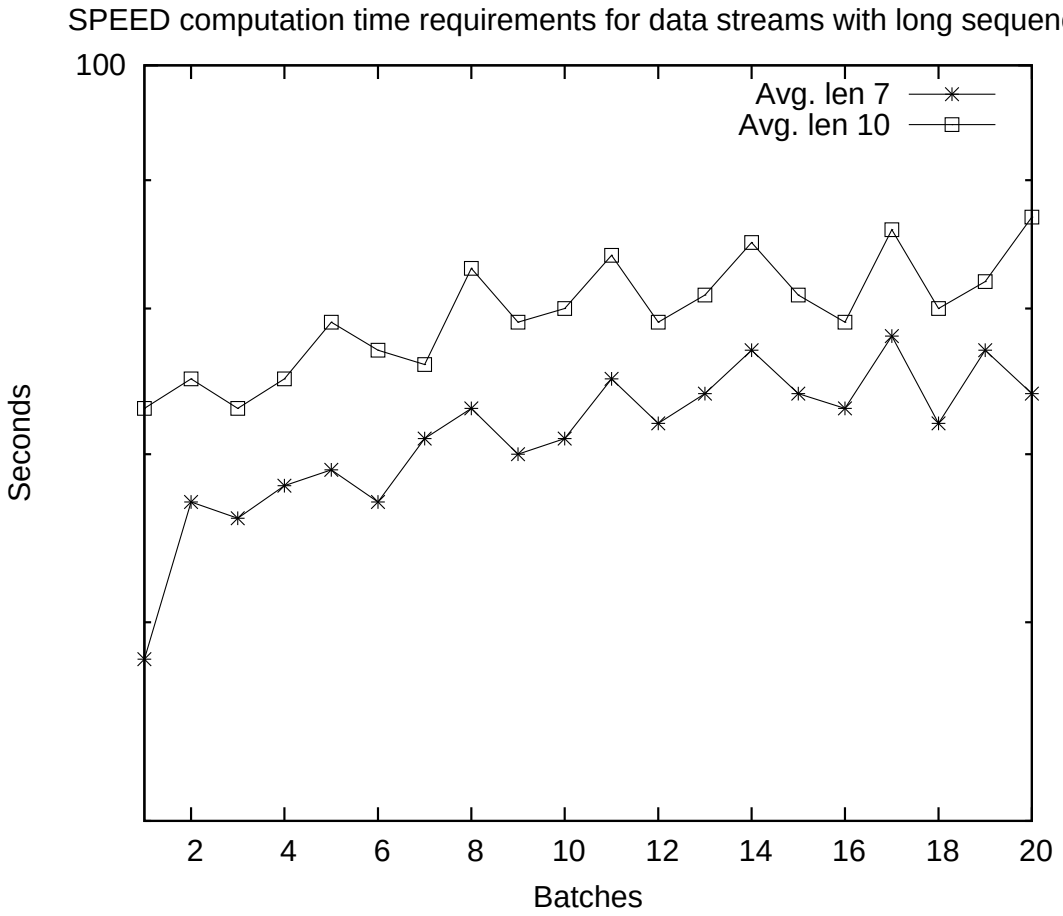


Fig.: Temps utilisé pour les longues séquences $\sigma = 0.1$ *et* $\epsilon = 0.075$

Conclusions et Perspectives

Conclusions et Perspectives

- Une nouvelle approche pour rechercher des motifs séquentiels dans les data streams
  - Une nouvelle représentation des itemsets
  - Une structure efficace pour rechercher et élaguer
  - La possibilité d'effectuer des requêtes sur des intervalles de temps arbitraires
- Améliorer les tilted-time windows pour stocker à la fois l'intervalle de temps et la distribution des items pendant la période (Processus stochastique)
- Traiter les longues séquences, i.e. de taille supérieure à celle d'un batch
- Définir un langage de requête approprié

Merci