Get_Move : fouille de données d'objets mobiles

Phan Nhat Hai [1,2], **Pascal Poncelet** [2], **Maguelonne Teisseire** [1]

1. TETIS - LIRMM 500 J. F. Breton 34093 Montpellier
{nhat-hai.phan,maguelonne.teisseire}@teledetection.fr

2. LIRMM - UM2 161 rue Ada 34095 Montpellier cedex 5
pascal.poncelet@lirmm.fr

RÉSUMÉ. *Les développements récents des techniques de géolocalisation ont généré de larges volumes de données associées aux objets mobiles. Une des tâches d'analyse de telles données reste à identifier les objets évoluant ensemble. Cette problématique peut être résolue par les motifs spatiotemporels et de nombreuses propositions ont été réalisées ces dernières années. Néanmoins chacune de ces approches se focalise sur un type de motif spécifique. Il est alors coûteux de vouloir tous les obtenir car il est nécessaire d'exécuter l'ensemble des algorithmes proposés. Pour répondre à ce problème, nous redéfinissons les motifs spatiotemporels dans le contexte des itemsets et proposons une approche unifiée, appelée GeT_Move, permettant d'extraire de tels motifs. Cet algorithme est proposé en deux versions dont l'une est incrémentale. Les expérimentations réalisées sur des données réelles et des données synthétiques soulignent l'efficacité de notre proposition qui surpasse les approches existantes.*

ABSTRACT. *Recent improvements in positioning technology has led to a much wider availability of massive moving object data. A crucial task is to find the moving objects that travel together. Usually, they are called spatio-temporal patterns. Due to the emergence of many different kinds of spatio-temporal patterns in recent years, different approaches have been proposed to extract them. However, each approach only focuses on mining a specific kind of pattern. In addition to the fact that it is a painstaking task due to the large number of algorithms used to mine and manage patterns, it is also time consuming. To address these issues, we first redefine spatio-temporal patterns in the itemset context. Secondly, we propose a unifying approach, named GeT_Move, using a frequent closed itemset-based spatio-temporal pattern-mining algorithm to mine and manage different spatio-temporal patterns. GeT_Move is implemented in two versions which are GeT_Move and Incremental GeT_Move. Experiments are performed on real and synthetic datasets and the experimental results show that our approaches are very effective and outperform existing algorithms in terms of efficiency.*

MOTS-CLÉS : *motif spatiotemporel, itemset clos fréquent, trajectoires.*

KEYWORDS: *spatio-temporal pattern, frequent closed itemset, trajectories.*

DOI:10.3166/ISI.18.4.145-169

1. Introduction

De nos jours, de nombreux appareils électroniques sont utilisés pour des applications du quotidien. Les GPS intégrés aux voitures ou associés aux animaux, les réseaux de capteurs et les téléphones portables permettent le suivi de presque n'importe quel type d'objet. Ceci conduit à un volume de plus en plus important de données numériques associées à de tels objets en mouvement. La communauté Fouille de données s'est donc attachée à trouver des modèles intéressants tels que l'étude du comportement animal, la planification des itinéraires et le contrôle des véhicules.

Les premières approches, visant à récupérer des informations à partir de bases de données spatiotemporelles, tentent de répondre à des requêtes concernant un prédicat simple ou celui du plus proche voisin. Par exemple, « trouver tous les objets se déplaçant à l'intérieur de la zone A entre 10h00-14h00 » ou « combien de voitures ont été conduites entre la place principale et l'aéroport le vendredi » (Romero, 2011). Les extensions des requêtes spatiales dans les outils et applications SIG permettent de répondre à ce type de requête. Cependant, ces techniques sont utilisées pour trouver la meilleure solution en explorant chaque objet spatial à un moment précis selon une mesure de distance (en général euclidienne). Il est néanmoins difficile de saisir des résultats obtenus sur le comportement collectif et les corrélations entre les entités concernées. C'est pourquoi, récemment, de nombreux motifs spatiotemporels ont été proposés (Gudmundsson, Kreveld, 2006 ; Jeung, Yiu *et al.*, 2008 ; Kalnis *et al.*, 2005 ; Li, Ding *et al.*, 2010 ; Verhein, 2009 ; Wang *et al.*, 2006 ; Cao *et al.*, 2006 ; Lee *et al.*, 2007 ; Mamoulis *et al.*, 2004 ; Hai *et al.*, 2012c ; Tang *et al.*, 2012 ; Zheng *et al.*, 2013).

Dans cet article, nous nous intéressons à l'interrogation des modèles qui capturent la notion de comportement « commun » entre les entités en mouvement. Il est particulièrement intéressant d'identifier les groupes d'objets en mouvement pour lesquels il existe une relation forte au sein d'une région spatiale et au cours d'une période donnée. Quelques exemples de tels modèles sont : les troupeaux (*flocks*) (Gudmundsson, Kreveld, 2006 ; Vieira *et al.*, 2009), les groupes mobiles (Kalnis *et al.*, 2005 ; Jensen *et al.*, 2007), les convois (Jeung, Yiu *et al.*, 2008 ; Jeung, Shen, Zhou, 2008), les essaims (swarms) clos (Li, Ding *et al.*, 2010 ; Li, Ji *et al.*, 2010), les essaims flous et clos (Hai *et al.*, 2012b), les motifs de groupe (Wang *et al.*, 2006), les motifs périodiques (Mamoulis *et al.*, 2004), etc.

Pour extraire l'ensemble de ces motifs, l'application des algorithmes est coûteuse en temps puisqu'elle nécessite des opérations consécutives. Il serait alors intéressant de disposer d'un unique algorithme qui pourrait extraire les différents types de motifs proposés. Les coûts de calcul seraient alors considérablement diminués et le processus grandement amélioré. C'est pourquoi nous proposons dans cet article de définir un algorithme permettant de mettre à disposition l'ensemble des motifs proposés en une seule exécution.

Dans la plupart des applications du monde réel, les emplacements des objets sont continuellement rapportés en utilisant le système de positionnement global (GPS). Par conséquent, de nouvelles données sont toujours disponibles. Si nous ne disposons pas

d'un algorithme incrémental, il sera nécessaire d'exécuter encore et encore les algorithmes sur la base de données entière. Ceci conduit bien sûr à un coût prohibitif et est chronophage. Un algorithme incrémental permet alors améliorer le processus en combinant les résultats issus des données existantes et les nouvelles données afin d'obtenir les résultats finaux. Dans ce contexte, nous proposons *Get_Move* : une approche unificatrice d'extraction de motifs spatiotemporels dont une des étapes se base sur un algorithme d'extraction d'itemsets fermés fréquents (FCI). Nos principales contributions sont les suivantes :

– nous redéfinissons l'extraction de motifs spatiotemporels dans le contexte de l'extraction d'itemsets, permettant ainsi d'extraire de façon efficace l'ensemble des motifs existants ;

– nous présentons les algorithmes, *Get_Move* et *Incremental_GeT_Move*, pour extraire efficacement les FCI à partir desquels les motifs spatiotemporels seront obtenus ;

– nous détaillons les résultats obtenus sur deux bases de données réelles et synthétiques. Les résultats montrent que nos techniques permettent d'extraire efficacement les différents types de motifs et qu'elles sont généralement plus efficaces que les algorithmes de référence ;

– le système développé est disponible en ligne [1] (Hai *et al.*, 2012a). L'utilisateur peut naviguer et comparer les différents motifs d'une manière simple et aisée.

La suite de cet article set organisée de la façon suivante. La section 2 présente les définitions préliminaires associées aux motifs spatiotemporels et l'état de l'art. Les définitions des motifs revisitées dans le contexte des itemsets sont proposées dans la section 3. Les algorithmes GeT_Move et Incremental_GeT_Move sont détaillés dans la section 4. Les expérimentations sont discutées dans la section 5. Enfin, la section 6 dresse le bilan et les perspectives associées à ce travail.

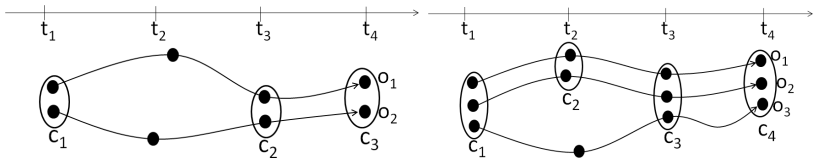
2. Motifs d'objets mobiles

Dans cette section, nous proposons un aperçu des principaux motifs spatiotemporels pour les objets mobiles et discutons des travaux connexes.

Tableau 1. Un exemple de base de données spatiotemporelles

Objet O_{DB}	Temps T_{DB}	x	y
o_1	t_1	2.3	1.2
o_2	t_1	2.1	1
o_1	t_2	10.3	28.1
o_2	t_2	0.3	1.2

1. http ://www.lirmm.fr/ ~phan/index.jsp



(a) Swarm (b) Convoy

Figure 1. Un exemple d'essaim et de convoi où c_1, c_2, c_3, c_4 sont des groupes d'objets évoluant ensemble pendant un intervalle

2.1. Définitions préliminaires

L'extraction de motifs spatiotemporels a été très étudiée ces dernières années. Ces motifs permettent de retracer les trajectoires similaires d'objets qui évoluent ensemble pendant un certain intervalle de temps. De nombreuses définitions ont été proposées telles que les *troupeaux* (*flocks*) (Gudmundsson, Kreveld, 2006) (Vieira *et al.*, 2009), les convois (Jeung, Yiu *et al.*, 2008 ; Jeung, Shen, Zhou, 2008), les *swarms*, les *swarms* clos (Li, Ding *et al.*, 2010 ; Li, Ji *et al.*, 2010), les groupes mobiles (Kalnis *et al.*, 2005 ; Jensen *et al.*, 2007), les motifs de groupe (Wang *et al.*, 2006) et *les motifs périodiques (Mamoulis* et al*., 2004)*.

Dans cet article, nous nous intéressons à proposer une méthode unificatrice permettant d'extraire l'ensemble des motifs proposés. Nous faisons l'hypothèse d'avoir à disposition un groupe d'objets mobiles $O_{DB} = \{o_1, o_2, \ldots, o_z\}$, un ensemble d'estampilles temporelles $T_{DB} = \{t_1, t_2, \ldots, t_n\}$ et pour chaque estampille temporelle $t_i \in T_{DB}$, les informations spatiales x, y pour chacun des objets de O_{DB} [2].

Le tableau 1 décrit un exemple de base de données spatiotemporelles. De façon générale, en fouille de données spatiotemporelles, nous nous intéressons à identifier des groupes d'objets restant ensemble sur une période. Nous noterons $O = \{o_{i_1}, o_{i_2}, \ldots, o_{i_p}\}(O \subseteq O_{DB})$ le groupe d'objets et $T = \{t_{a_1}, t_{a_2}, \ldots, t_{a_m}\}(T \subseteq T_{DB})$ les estampilles temporelles pendant lesquelles les objets restent ensemble. Soit ε un support minimum défini par l'utilisateur représentant le nombre minimum d'objets devant être ensemble et min_t le nombre minimum d'estampilles temporelles le vérifiant. Alors, $|O|$ (resp. $|T|$) doit être supérieur ou égal à ε (resp. min_t). Nous allons maintenant définir plus formellement les différents types de motifs.

Intuitivement, un *essaim* est un groupe d'objets O contenant au moins ε éléments qui sont proches les uns des autres pour au moins min_t estampilles temporelles. Il peut être défini de la façon suivante :

2. Les informations spatiales peuvent être par exemple la localisation GPS.

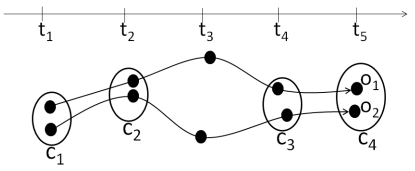


Figure 2. Un exemple de motif de groupe

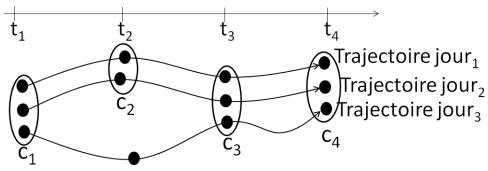


Figure 3. Un exemple de motif périodique

DEFINITION 1. — *Swarm (Li, Ding* et al.*, 2010). Un couple* (O, T) *est un swarm si :*

$$\begin{cases} (1) : \forall t_{a_i} \in T, \exists c \textit{ tel que } O \subseteq c, \textit{ c est un cluster.} \\ \textit{Il y a au moins un cluster contenant} \\ \textit{tout objet de O à chaque estampille temporelle de } T. \\ (2) : |O| \geq \varepsilon. \\ \textit{Il y a au moins } \varepsilon \textit{ objets.} \\ (3) : |T| \geq min_t. \\ \textit{Il y a au moins } min_t \textit{ estampilles.} \end{cases} \quad (1)$$

Comme illustré figure 1, si $\varepsilon = 2$ et $min_t = 2$, nous pouvons trouver les *swarms* suivants : $(\{o_1, o_2\}, \{t_1, t_3\})$, $(\{o_1, o_2\}, \{t_1, t_4\})$, $(\{o_1, o_2\}, \{t_3, t_4\})$, $(\{o_1, o_2\}, \{t_1, t_3, t_4\})$. Nous pouvons noter que ces groupes sont en fait redondants puisqu'ils peuvent être réunis dans le swarm $(\{o_1, o_2\}, \{t_1, t_3, t_4\})$.

Pour éviter ce problème, (Li, Ding *et al.*, 2010) ont proposé la notion de *closed swarm*. Un swarm (O, T) est objet-fermé si en fixant T, O ne peut pas être augmenté. De façon similaire, un *swarm* (O, T) est *temporel-fermé* si en fixant O, T ne peut pas être augmenté. Un *swarm* (O, T) est un essaim clos s'il est à la fois objet-fermé et temporel-fermé. Ceci nous amène à la définition suivante :

DEFINITION 2. — *Swarm clos (Li, Ding* et al.*, 2010). Un couple* (O, T) *est un closed swarm si :*

$$\begin{cases} (1) : (O, T) \textit{ est un swarm.} \\ (2) : \nexists O' \textit{ tel que } (O', T) \textit{ est un swarm et } O \subset O'. \\ (3) : \nexists T' \textit{ tel que } (O, T') \textit{ est un swarm et } T \subset T'. \end{cases} \quad (2)$$

Dans l'exemple précédent, $(\{o_1, o_2\}, \{t_1, t_3, t_4\})$ est un *closed swarm*.

Un *convoi* est également un groupe d'objets tel que les objets sont proches les uns des autres pendant au moins min_t estampilles temporelles. Dans ce cas, les estampilles doivent être consécutives contrairement au swarm.

DEFINITION 3. — *Convoi (Jeung, Yiu* et al.*, 2008). Un couple* (O, T)*, est un convoi si :*

$$\begin{cases} (1) : (O,T) \textit{ est un swarm.} \\ (2) : \forall i, 1 \leq i < |T|, \ t_{a_i}, t_{a_{i+1}} \textit{ sont consecutives.} \end{cases} \quad (3)$$

Par exemple, sur la figure 1b, avec $\varepsilon = 2$ et $min_t = 2$, nous obtenons deux convois $(\{o_1, o_2\}, \{t_1, t_2, t_3, t_4\})$ et $(\{o_1, o_2, o_3\}, \{t_3, t_4\})$.

Jusqu'à présent, nous avons considéré un groupe d'objets qui se déplacent à proximité les uns aux autres pour un long intervalle. Par exemple, comme le montrent (Han *et al.*, 2010), ces motifs ont pratiquement la même définition. Fondamentalement, la principale différence repose sur les techniques de clustering utilisées. Pour extraire les *flocks*, il faut considérer une définition rigide du rayon alors que les groupes mobiles et les convois peuvent adopter un algorithme de clustering basé sur la densité (par exemple DBScan (Ester *et al.*, 1996)). Les clusters mobiles peuvent être considérés comme des cas particuliers de convois avec la condition supplémentaire de partage des objets entre deux estampilles temporelles consécutives (Han *et al.*, 2010). Par conséquent, dans la suite de cet article, dans un souci de concision et de clarté, nous nous concentrons sur les concepts de convoi et les algorithmes de regroupement fondé sur la densité.

Selon les définitions précédentes, la différence entre les convois et les *swarms* repose sur l'aspect consécutif ou non consécutif des intervalles temporels. (Wang *et al.*, 2006) proposent un modèle général, appelé *motif de groupe (group pattern)*, qui est une combinaison des deux définitions, les convois et les swarms clos. Le motif de groupe est ainsi un ensemble de convois disjoints qui sont générés pour un même groupe d'objets dans des intervalles de temps différents. En considérant un convoi comme un point de temps, un motif de groupe peut être considéré comme un *swarm* de convois disjoints. En outre, le motif de groupe ne peut pas être agrandi en termes d'objets et en nombre de convois. Par conséquent, le motif de groupe est essentiellement un *swarm* clos de convois disjoints. Formellement, le motif de groupe peut être défini comme suit :

DEFINITION 4. — *Motif de groupe (Wang* et al.*, 2006). Etant donné un ensemble d'objets* O*, un poids minimum* min_{wei}*, un ensemble de convois disjoints* $T_{\mathcal{S}} = \{s_1, s_2, \ldots, s_n\}$*, un nombre de convoi minimum* min_c*.* $(O, T_{\mathcal{S}})$ *est un motif de groupe si :*

$$\begin{cases} (1) : (O,T_{\mathcal{S}}) \textit{ est un closed swarm respectant } \varepsilon, min_c. \\ (2) : \frac{\sum_{i=1}^{|T_{\mathcal{S}}|} |s_i|}{|T_{DB}|} \geq min_{wei}. \end{cases} \quad (4)$$

Notons que min_c *est uniquement appliqué pour* $T_{\mathcal{S}}$ *(e.g.* $|T_{\mathcal{S}}| \geq min_c$*).*

Pour illustration, à partir de l'exemple de la figure 2, avec $min_t = 2$ et $\varepsilon = 2$, nous obtenons un ensemble de convois $T_{\mathcal{S}} = \{(\{o_1, o_2\}, \{t_1, t_2\}), (\{o_1, o_2\}, \{t_4, t_5\})\}$. De plus, avec $min_c = 1$, $(\{o_1, o_2\}, T_{\mathcal{S}})$ est un essaim clos de convois car $|T_{\mathcal{S}}| = 2 \geq min_c$, $|O| \geq \varepsilon$ et $(O, T_{\mathcal{S}})$ ne peut pas être augmenté. Enfin, avec $min_{wei} = 0.5$, $(O, T_{\mathcal{S}})$ est un motif de groupe puisque $\frac{|[t_1,t_2]|+|[t_4,t_5]|}{|T_{DB}|} = \frac{4}{5} \geq min_{wei}$.

Nous nous sommes intéressés jusqu'à présent aux motifs de groupe d'objets alors qu'extraire des motifs propres aux objets pris comme des individus isolés est également pertinent. (Mamoulis *et al.*, 2004) proposent le concept de *motifs périodiques (periodic patterns)* exprimant les routes (approximativement) suivies par les objets à des intervalles de temps réguliers. Par exemple, chaque jour des personnes se lèvent à la même heure et suivent la même route jusqu'à leur travail. De façon informelle, étant donnée une trajectoire d'objets de $\mathcal{N}$ points, $\mathcal{T}_\mathcal{P}$ est le nombre d'estampilles temporelles pour lesquelles un motif peut s'instancier de nouveau. Une trajectoire d'objets est décomposée en $\lfloor \frac{\mathcal{N}}{\mathcal{T}_\mathcal{P}} \rfloor$ sous-trajectoires. $\mathcal{T}_\mathcal{P}$ dépend des données et n'a pas de valeur prédéfinie. Par exemple, $\mathcal{T}_\mathcal{P}$ peut être égal à une journée pour les applications gérant le traffic routier puisque les voitures ont en général des motifs journaliers ou à une année pour les études sur la migration des animaux. Pour illustration, figure 3, il est possible de voir des trajectoires d'objets décomposées en sous-trajectoires journalières.

Un motif périodique est un essaim clos extrait avec $\lfloor \frac{\mathcal{N}}{\mathcal{T}_\mathcal{P}} \rfloor$ sous-trajectoires. C'est ainsi que sur la figure 3, nous avons 3 sous-trajectoires journalières dont nous extrayons les deux motifs périodiques suivants : $\{c_1, c_2, c_3, c_4\}$ et $\{c_1, c_3, c_4\}$. La différence principale pour les motifs périodiques réside dans le prétraitement des données mais la définition reste similaire à celle des swarms clos.

2.2. Etat de l'art

Comme nous l'avons mentionné précédemment, de nombreuses approches ont été proposées pour l'extraction de motifs. Le lecteur intéressé peut se reporter à (Bogorny, Shekhar, 2010) et (Han *et al.*, 2010) qui décrivent les motifs intéressants et les approches les plus efficaces. Par exemple, (Gudmundsson, Kreveld, 2006) et (Vieira *et al.*, 2009) définissent les motifs de troupeau pour lesquels les mêmes objets évoluent ensemble dans un espace circulatoire défini selon un rayon, (Kalnis *et al.*, 2005) proposent la notion de groupe mobile, alors que (Jeung, Yiu *et al.*, 2008) définissent les motifs de convoi.

(Jeung, Yiu *et al.*, 2008) adoptent l'algorithme DBScan (Ester *et al.*, 1996) pour trouver les motifs convois candidats. Les auteurs proposent trois algorithmes qui intègrent des techniques de simplification des trajectoires en première étape. Les mesures de distance sont calculées sur des segments de trajectoires en opposition à des mesures de distance entre points. Un autre problème est la représentation de ces trajectoires car certaines ne possèdent pas les relevés aux mêmes estampilles ou intervalles temporels. Dans ce cas, les mesures de densité ne sont pas applicables. Pour résoudre le problème des absences de relevés, les auteurs interpolent les trajectoires réelles avec des points virtuels et peuvent alors appliquer les mesures de densité sur les segments de trajectoire. Le convoi est alors défini comme un candidat lorsqu'il a au moins *k* groupes durant *n* estampilles temporelles consécutives.

Récemment, (Li, Ding *et al.*, 2010) ont proposé les concepts d'essaim (swarm) et d'essaim clos. L'algorithme est basé sur une approche « ObjectGrowth » qui est une démarche en profondeur basée sur l'espace de recherche des ensembles d'objets

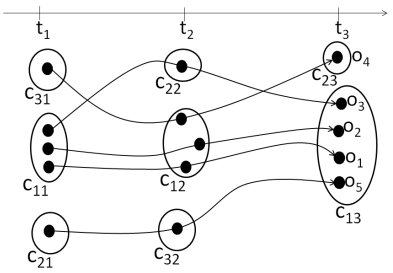


Figure 4. Un exemple de groupe d'objets mobiles

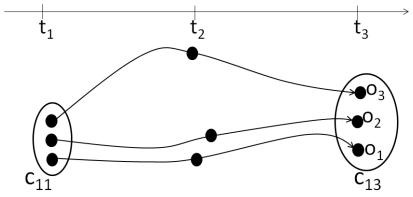


Figure 5. Un essaim extrait de l'exemple de la figure 4

(c'est-à-dire les sous-ensembles de O_{DB}). Pour cela, l'espace de recherche est parcouru en « profondeur d'abord » dans l'arbre des sous-ensembles de O_{DB}, préordonnés. Néanmoins, l'énumération des ensembles d'objets reste complexe en $O\ (2^{|O_{DB}|})$ et afin d'améliorer le processus de parcours, les auteurs proposent deux règles d'optimisation. La première règle, appelée *Apriori Pruning*, permet de limiter le parcours du sous-arbre lorsqu'il y a des traverses ne vérifiant pas min_t. La seconde règle, appelée *Backward Pruning*, utilise la propriété de fermeture. S'il existe un sur-ensemble de l'ensemble d'objets examinés ayant les mêmes estampilles temporelles, dans ce cas, le sous-arbre examiné est inutile. Après avoir éliminé ces candidats invalides, les candidats restant ne sont pas forcément clos. Il est alors nécessaire de les vérifier afin de n'extraire que les essaims clos.

(Wang *et al.*, 2006) proposent deux algorithmes pour extraire des motifs de groupe (*group patterns*), connus sous le nom de *Apriori-like Group Pattern mining algorithm* et *Valid Group-Growth algorithm*. Le premier utilise les propriétés de Apriori pour extraire des motifs de groupe, il s'agit donc d'une extension de l'algolithme Apriori de (Agrawal, Srikant, 1994). Le deuxième est basé sur une idée similaire à l'algorithme FP-growth proposé dans (Han *et al.*, 2000). Récemment, (Romero, 2011) a défini une approche basée sur les itemsets pour extraire des motifs de troupeau (*flock*).

Même si ces approches sont efficaces, elles souffrent toutes du problème d'individualité, c'est-à-dire de n'extraire que les motifs pour lesquels elles ont été définies. Il est difficile pour un expert décideur de connaître à l'avance le type de motifs qui sera le plus pertinent à extraire des données sur lesquelles il travaille. L'objet de notre proposition est une approche automatique permettant d'extraire simplement et efficacement différents types de motifs spatiotemporels. Cette proposition est détaillée dans les sections suivantes.

Tableau 2. Matrice de clusters

T_{DB}		t_1			t_2			t_3	
Clusters C_{DB}		c_{11}	c_{21}	c_{31}	c_{12}	c_{22}	c_{32}	c_{13}	c_{23}
O_{DB}	o_1	1			1			1	
	o_2	1			1			1	
	o_3	1				1		1	
	o_4			1	1				1
	o_5		1				1	1	

3. Motifs spatiotemporels dans le contexte des itemsets

Dans cette section, nous proposons de définir une approche unificatrice permettant d'extraire et de gérer différents types de motifs.

Les motifs peuvent être considérés comme des clusters évoluant au cours du temps. L'analyse des corrélations entre clusters permet d'obtenir des informations complémentaires. Ainsi, si certains clusters partagent des caractéristiques communes (les mêmes objets), cela peut correspondre à un motif. Par conséquent, si un cluster est considéré comme un item, nous aurons un ensemble d'items (nommé itemset). La problématique consiste alors à combiner de façon efficace les items (clusters) pour trouver les itemsets (un ensemble de clusters) qui partagent certaines caractéristiques ou satisfont des propriétés permettant de les considérer comme des motifs. Pour décrire les évolutions de clusters, les données spatiotemporelles sont présentées comme une matrice de clusters à partir desquels les motifs vont pouvoir être extraits.

DEFINITION 5. — *Matrice de clusters. Supposons un ensemble de clusters* $C_{DB} = \{C_1, C_2, \ldots, C_n\}$ *avec* $C_i = \{c_{i_1 t_i}, c_{i_2 t_i}, \ldots, c_{i_m t_i}\}$ *un ensemble de clusters à l'estampille temporelle* t_i*, une matrice de clusters est une matrice de taille* $|O_{DB}| \times |C_{DB}|$*. Chaque ligne de la matrice correspond à un objet et chaque colonne représente un cluster. La valeur de la cellule de la matrice* (o_i, c_j) *est égale à 1 (respectivement 0) si* o_i *est dans (resp. n'est pas dans) le cluster* c_j*.*

Dans le tableau 2, les données de la figure 4 sont organisées selon la matrice de clusters. L'objet o_1 appartient au cluster c_{11} à la date t_1. Dans la suite de cet article, nous utiliserons c_{ij} pour représenter le cluster c_i à la date t_j. La cellule de la matrice (o_1-c_{11}) vaut donc 1 alors que la cellule (o_4-c_{11}) est vide car l'object o_4 n'appartient pas au cluster c_{11}.

En représentant les données à l'aide d'une matrice de clusters, chaque objet intervient comme une transaction et chaque cluster c_j représente un item. Un itemset est ainsi formé par $\Upsilon = \{c_{t_{a_1}}, c_{t_{a_2}}, \ldots, c_{t_{a_p}}\}$ aux dates $T_\Upsilon = \{t_{a_1}, t_{a_2}, \ldots, t_{a_p}\}$ avec $t_{a_1} < t_{a_2} < \ldots < t_{a_p}$, $\forall a_i : t_{a_i} \in T_{DB}, c_{t_{a_i}} \in C_{a_i}$. Le support de l'itemset Υ, noté $\sigma(\Upsilon)$, est le nombre d'objets communs des items participant à Υ, $O(\Upsilon) = \bigcap_{i=1}^{p} c_{t_{a_i}}$. De plus, la taille de Υ, notée $|\Upsilon|$, est le nombre d'item ou de date ($= |T_\Upsilon|$).

Tableau 2, $\Upsilon = \{c_{11}, c_{12}\}$ verifie $\sigma(\Upsilon) = 2$. Chaque item (resp. cluster) de Υ, c_{11} et c_{12}, apparaissent dans les transactions (resp. objet) o_1, o_2. La taille de $|\Upsilon|$ est le nombre d'item ($= 2$).

Même si le nombre de clusters peut être grand, la taille maximale des itemsets est limitée à $|T_{DB}|$ grâce à l'algorithme de clustering utilisé qui est basé sur la densité des groupes, car deux clusters à la même date ne peuvent pas être dans le même itemset.

Maintenant, nous allons définir les propriétés propres aux itemsets qui vont nous permettre d'extraire les motifs présentés à la section 2.

PROPRIÉTÉ.— *Swarm.* Etant donné un itemset fréquent $\Upsilon = \{c_{t_{a_1}}, c_{t_{a_2}}, \ldots, c_{t_{a_p}}\}$, $(O(\Upsilon), T_\Upsilon)$ est un swarm si et seulement si :

$$\begin{cases} (1) : \sigma(\Upsilon) \geq \varepsilon \\ (2) : |\Upsilon| \geq min_t \end{cases} \tag{5}$$

PREUVE. Par définition, nous avons $\sigma(\Upsilon) \geq \varepsilon$ et $\sigma(\Upsilon) = |O(\Upsilon)|$ donc $|O(\Upsilon)| \geq \varepsilon$. Comme $|\Upsilon| \geq min_t$ et $|\Upsilon| = |T_\Upsilon|$ alors $|T_\Upsilon| \geq min_t$. De plus, $\forall t_{a_j} \in T_\Upsilon, O(\Upsilon) \subseteq c_{t_{a_j}}$, ceci signifie qu'à chaque date, il y a un cluster contenant tous les objets de $O(\Upsilon)$. Donc $(O(\Upsilon), T_\Upsilon)$ est un swarm car il satisfait toutes les conditions de la *définition 1*.

Par exemple, figure 5, à partir de l'itemset fréquent $\Upsilon = \{c_{11}, c_{13}\}$, en supposant le support minimal $\varepsilon = 2$ et $min_t = 2$, nous avons $(O(\Upsilon) = \{o_1, o_2, o_3\}, T_\Upsilon = \{t_1, t_3\})$ qui est un swarm car $\sigma(\Upsilon) = 3 > \varepsilon$ et $|\Upsilon| = 2 \geq min_t$.

Un essaim clos se définit à partir des conditions d'extension des objets et des dates comme explicité dans la propriété suivante.

PROPRIÉTÉ.— *Closed Swarm.* Etant donné un itemset fréquent $\Upsilon = \{c_{t_{a_1}}, c_{t_{a_2}}, \ldots, c_{t_{a_p}}\}$. $(O(\Upsilon), T_\Upsilon)$ est un essaim clos si et seulement si :

$$\begin{cases} (1) : (O(\Upsilon), T_\Upsilon) \textit{ est un swarm.} \\ (2) : \nexists \Upsilon' \textit{ tel que } O(\Upsilon) \subset O(\Upsilon'), T_{\Upsilon'} = T_\Upsilon \textit{ et} \\ (O(\Upsilon'), T_\Upsilon) \textit{ est un swarm.} \\ (3) : \nexists \Upsilon' \textit{ tel que } O(\Upsilon') = O(\Upsilon), T_\Upsilon \subset T_{\Upsilon'} \textit{ et} \\ (O(\Upsilon), T_{\Upsilon'}) \textit{ est un swarm.} \end{cases} \tag{6}$$

PREUVE. Soit $(O(\Upsilon), T_\Upsilon)$ un swarm, si $\nexists \Upsilon'$ tel que $O(\Upsilon) \subset O(\Upsilon'), T_{\Upsilon'} = T_\Upsilon$, $(O(\Upsilon'), T_\Upsilon)$ est un swarm et $(O(\Upsilon), T_\Upsilon)$ ne peut pas être étendu en termes d'objets. De plus, si $\nexists \Upsilon'$ tel que $O(\Upsilon') = O(\Upsilon), T_\Upsilon \subset T_{\Upsilon'}$ et $(O(\Upsilon), T_{\Upsilon'})$ est un swarm alors $(O(\Upsilon), T_\Upsilon)$ ne peut pas être étendu en termes de date. Par conséquent, $(O(\Upsilon), T_\Upsilon)$ est un swarm qui ne peut pas être étendu en termes d'objets et en termes de date, donc $(O(\Upsilon), T_\Upsilon)$ est un essaim clos selon la *définition 2*.

Selon la *définition 3*, un convoi est un swarm qui satisfait la condition des dates consécutives. Donc, à partir d'un itemset, il est possible d'extraire un convoi si la propriété suivante est vérifiée.

PROPRIÉTÉ.— *Convoi.* Etant donné un itemset fréquent $\Upsilon = \{c_{t_{a_1}}, c_{t_{a_2}}, \ldots, c_{t_{a_p}}\}$. $(O(\Upsilon), T_\Upsilon)$ est un convoi si et seulement si :

$$\begin{cases} (1) : (O(\Upsilon), T_\Upsilon) \textit{ est un swarm.} \\ (2) : \forall j, 1 \leq j < p : t_{a_{j+1}} = t_{a_j} + 1. \end{cases} \tag{7}$$

PREUVE. Soit $(O(\Upsilon), T_\Upsilon)$ un swarm, si Υ vérifie la condition (2), cela signifie que les dates de Υ sont consecutives. Donc, $(O(\Upsilon), T_\Upsilon)$ est un convoi selon la *définition 3*.

Nous ne détaillons pas les propriétés et les preuves pour les moving clusters qui sont une extension des *propriétés 3*. Un moving cluster (Kalnis *et al.*, 2005) doit partager les mêmes objets entre deux dates afin de vérifier la proportion d'intégrité. En ce qui concerne les motifs périodiques, la propriété est similaire à la *propriété 3* avec une légère modification dans la construction de la matrice de cluster telle que « chaque o puisse être une sous-trajectoire ».

Un motif de groupe est un ensemble disjoint de convois qui partagent les mêmes objets mais à des dates différentes. Dans ce cas, la propriété des motifs de groupe se décrit de la façon suivante.

PROPRIÉTÉ.— *Motif de groupe.* Etant donné un itemset fréquent $\Upsilon = \{c_{t_{a_1}}, c_{t_{a_2}}, \ldots, c_{t_{a_p}}\}$, un poids minimum min_{wei}, un nombre minimum de convoi min_c, un ensemble de segments temporels consécutifs $T_\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$. $(O(\Upsilon), T_\mathcal{S})$ est un motif de groupe si et seulement si :

$$\begin{cases} (1) : |T_\mathcal{S}| \geq min_c. \\ (2) : \forall s_i, s_i \subseteq T_\Upsilon, |s_i| \geq min_t. \\ (3) : \bigcap_{i=1}^{n} s_i = \emptyset, \bigcap_{i=1}^{n} O(s_i) = O(\Upsilon). \\ (4) : \forall s \notin T_\mathcal{S}, s \textit{ est un convoi, } O(\Upsilon) \not\subseteq O(s). \\ (5) : \frac{\sum_{i=1}^{n} |s_i|}{|T|} \geq min_{wei}. \end{cases} \tag{8}$$

PREUVE. Si $|T_\mathcal{S}| \geq min_c$ alors nous avons au moins min_c intervalles temporels consécutifs $s_i \in T_\mathcal{S}$. De plus, si $\forall s_i, s_i \subseteq T_\Upsilon$, alors $O(\Upsilon) \subseteq O(s_i)$. Si $|s_i| \geq min_t$ alors $(O(\Upsilon), s_i)$ est un convoi (*Définition 3*). $T_\mathcal{S}$ est un ensemble de convois de $O(\Upsilon)$ et si $\bigcap_{i=1}^{n} s_i = \emptyset$ alors $T_\mathcal{S}$ est un ensemble disjoint de convois. Si $\forall s \notin T_\mathcal{S}, s$ est un convoi et si $O(\Upsilon) \not\subseteq O(s)$ alors $\nexists T_{\mathcal{S}'}$ tel que $T_\mathcal{S} \subset T_{\mathcal{S}'}$ et $\bigcap_{i=1}^{|T_{\mathcal{S}'}|} O(s_i) = O(\Upsilon)$. Ce qui signifie que $(O(\Upsilon), T_\mathcal{S})$ ne peut pas être étendu en termes de *nombre de convois*. De façon similaire, si $\bigcap_{i=1}^{n} O(s_i) = O(\Upsilon)$ alors $(O(\Upsilon), T_\mathcal{S})$ ne peut pas être étendu en termes d'*objets*. Par conséquent, $(O(\Upsilon), T_\mathcal{S})$ est un essaim clos de convois disjoints car $|O(\Upsilon)| \geq \varepsilon, |T_\mathcal{S}| \geq min_c$ et $(O(\Upsilon), T_\mathcal{S})$ ne peut pas être étendu (*Definition 2*). Enfin, si $(O(\Upsilon), T_\mathcal{S})$ vérifie la condition (5) alors il s'agit d'un motif de groupe selon la *définition 4*.

Nous avons présenté un ensemble de propriétés permettant de définir des motifs spatiotemporels à partir d'itemsets fréquents. Nous allons désormais nous intéresser à la définition d'algorithmes d'extraction d'itemsets afin d'extraire l'ensemble des

motifs spatiotemporels ainsi définis. Nous nous intéressons tout d'abord à la phase d'extraction de swarm comme le précise le lemme suivant.

LEMME 6. — *Soit $FI = \{\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_l\}$ l'ensemble des itemsets fréquents extraits à partir de la matrice de cluster avec $minsup = \varepsilon$, tous les swarms (O, T) peuvent être extraits de FI.*

PREUVE. Soit (O, T) un swarm avec $T = \{t_{a_1}, t_{a_2}, \ldots, t_{a_m}\}$, selon la *définition 1*, $|O| \geq \varepsilon$. Comme (O, T) est un swarm alors $\forall t_{a_i} \in T, \exists c_{t_{a_i}}$ tel que $O \subseteq c_{t_{a_i}}$ et donc $\bigcap_{i=1}^{m} c_{t_{a_i}} = O$. De plus, nous savons que $\forall c_{t_{a_i}}$, $c_{t_{a_i}}$ est un item, $\exists \Upsilon = \bigcup_{i=1}^{m} c_{t_{a_i}}$ est un itemset et $O(\Upsilon) = \bigcap_{i=1}^{m} c_{t_{a_i}} = O, T_\Upsilon = \bigcup_{i=1}^{m} t_{a_i} = T$. $(O(\Upsilon), T_\Upsilon)$ est un swarm et (O, T) est obtenu à partir de Υ. De plus, $\sigma(\Upsilon) = |O(\Upsilon)| = |O| \geq \varepsilon$ et Υ est un itemset fréquent et $\Upsilon \in FI$. Finalement, $\forall (O, T)$ tel que si (O, T) est un swarm tel que $\exists \Upsilon$ tel que $\Upsilon \in FI$ alors (O, T) peut être extrait de Υ, nous pouvons alors conclure que tout swarm (O, T) peut être extrait de FI.

En ajoutant des contraintes telles que les dates consécutives, la non-extension temporelle ou d'objet, l'intégrité de proportion aux swarms, nous pouvons extraire les convois, les swarms clos et les groupes mobiles. Ainsi, selon le *lemme 6*, nous pouvons extraire ces motifs à partir des itemsets fréquents. Il en est de même pour les motifs périodiques qui sont similaires aux swarms clos. Nous allons nous intéresser maintenant à l'extraction des motifs de groupe à partir des itemsets fréquents.

LEMME 7. — *Soit $FI = \{\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_l\}$ l'ensemble des itemsets fréquents extraits à partir de la matrice de cluster avec $minsup = \varepsilon$, les motifs de groupe $(O, T_\mathcal{S})$ peuvent être extraits à partir de FI.*

PREUVE. $\forall (O, T_\mathcal{S})$ est un motif de groupe, nous avons $\exists T_\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ et $T_\mathcal{S}$ est un ensemble disjoint de convois O. (O, T_{s_i}) est un convoi et $\forall s_i \in T_\mathcal{S}, \forall t \in T_{s_i}, \exists c_t$ tel que $O \subseteq c_t$. Supposons que C_{s_i} est un ensemble de clusters correspondant à s_i, alors $\exists \Upsilon$, Υ est un itemset, $\Upsilon = \bigcup_{i=1}^{n} C_{s_i}$ et $O(\Upsilon) = \bigcap_{i=1}^{n} O(C_{s_i}) = O$. De plus, $(O, T_\mathcal{S})$ est un motif de groupe donc $|O| \geq \varepsilon$ donc $|O(\Upsilon)| \geq \varepsilon$. Υ est un itemset frequent et $\Upsilon \in FI$ car Υ est un itemset et $\sigma(\Upsilon) = |O(\Upsilon)| \geq \varepsilon$. Par conséquent, $\forall (O, T_\mathcal{S}), \exists \Upsilon \in FI$ tel que $(O, T_\mathcal{S})$ pouvant être extrait de Υ et donc tout motif de groupe peut être extrait de FI.

4. Algorithmes d'extraction de motifs spatiotemporels

Dans cette section, nous proposons deux approches, *GeT_Move* et *Incremental GeT_Move*, et les algorithmes associés, pour extraire efficacement les motifs spatiotemporels. La figure 6 illustre le processus global adopté.

Lors de la première étape, une approche de clustering est appliquée pour chaque estampille temporelle afin de regrouper des objets en différents groupes. Pour chaque estampille, t_a, nous avons un ensemble de cluster $C_a = \{c_{1t_a}, c_{2t_a}, \ldots, c_{mt_a}\}$, avec $1 \leq k \leq m, c_{kt_a} \subseteq O_{DB}$. Les données spatiotemporelles peuvent alors être structurées en une matrice de cluster CM.

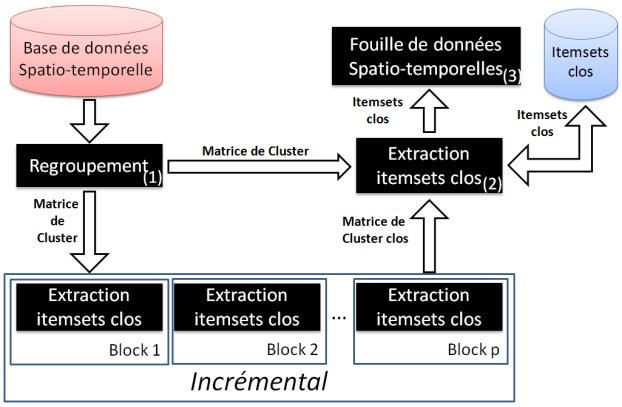


Figure 6. Le processus global

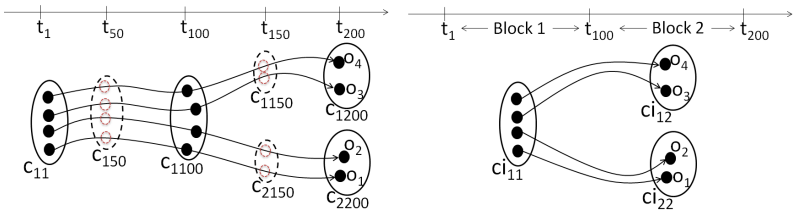
4.1. GeT_Move

Après avoir généré la matrice de cluster CM, un algorithme d'extraction d'itemsets clos est appliqué sur la matrice CM. En appliquant les propriétés sur les itemsest obtenus, il est possible d'obtenir les motifs souhaités.

Dans cet article, nous adoptons l'algorithme LCM (Uno *et al.*, 2004) pour extraire les itemsets fréquents et clos (ICF). Dans le processus, certains candidats inutiles peuvent être supprimés. Pour les motifs spatiotemporels, les items (resp. cluster) doivent avoir des estampilles temporelles différentes et donc les itemsets (resp. clusters) qui forment un ICF doivent être à des dates différentes. Par conséquent, les ICF qui comprennent plus d'un item à la même date sont rejetés.

En fonction de ces éléments, nous connaissons la longueur maximum possible des ICF qui est le nombre d'estampille temporelle de la base de données $|T_{DB}|$. De plus, l'espace de recherche parcouru par LCM dépend uniquement du nombre d'objets (les transactions) $|O_{DB}|$ et de la taille maximum des itemsets $|T_{DB}|$. Par conséquent, en adoptant LCM et les propriétés énoncées, les performances de *GeT_Move* ne sont pas affectées par le nombre de cluster et les temps de calcul sont donc considérablement réduits.

L'algorithme 1 détaille *GeT_Move*. Son noyau est l'algorithme LCM qui a été légèrement modifié pour y ajouter la règle d'élagage proposée et l'extraction des motifs à partir des ICF. L'ensemble des ICF X est vide et initialisé avec l'item i (lignes 2-3). En ajoutant i dans X, nous avons $X[i]$ et si $X[i]$ est un ICF alors $X[i]$ est utilisé comme un générateur d'un nouvel ICF, par l'appel à LCM_Iter($X, \mathcal{T}(X), i(X)$) (lignes 4-5). Avec LCM_Iter, nous testons tout d'abord la propriété de la section 3 (ligne 8) pour le ICF X. Ensuite, pour chaque transaction $t \in \mathcal{T}(X)$, nous ajoutons



(a) La base de données. (b) Données après ICF sur les blocs.

Figure 7. (a) Un exemple de base de données (b) $ci_{11}, ci_{12}, ci_{22}$ sont des ICF extraits des blocs 1 et 2

tous les items j qui sont plus grands que $i(X)$ et satisfont la règle d'élagage dans l'ensemble des occurrences $\mathcal{J}[j]$ (lignes 9-11). Puis, pour chaque $j \in \mathcal{J}[j]$, nous évaluons si $\mathcal{J}[j]$ est un ICF. Si tel est le cas, nous appelons de nouveau LCM_Iter avec ce nouveau générateur (lignes 12-14). Ensuite, pour les sous-fonctions d'extraction de motifs (lignes 16-37), l'algorithme vérifie simplement les propriétés sur les itemsets X pour sélectionner les motifs spatiotemporels [3].

4.2. Incremental GeT_Move

De façon générale, la taille des transactions peut être importante selon $|T_{DB}|$. Tout nouvel ajout d'information conduit à cette situation. Dans ce cas, l'extraction des IFC peut être ralentie du fait des longues transactions ainsi obtenues. Il y a donc un réel enjeu à ne pas appliquer de nouveau *GeT_Move* sur la base de données dans sa globalité, sous peine d'avoir une approche trop consommatrice en temps d'exécution.

C'est pourquoi, nous proposons l'algorithme *Incremental GeT_Move*. L'idée de base est de limiter la taille des transactions en divisant les trajectoires (resp. la matrice de cluster CM) à des intervalles plus courts, appelés blocs. En appliquant l'extraction des ICF sur chacun des intervalles, il est ensuite possible de compresser les données par des ICF locaux. De cette façon, la longueur des itemsets et le nombre d'items peuvent être considérablement réduits.

Par exemple, figure 7, si nous considérons $[t_1, t_{100}]$ comme un bloc et $[t_{101}, t_{200}]$ un autre bloc, la longueur maximale des itemsets dans les deux blocs est de 100 (au lieu de 200). Ainsi les données d'origine sont compressées (e.g. figure 7b) et seuls les 3 items $ci_{11}, ci_{12}, ci_{22}$ apparaissent.

3. Avec X un itemset, $X[i] := X \cup i, i(X)$ est le dernier item de $X, \mathcal{T}(X)$ est la liste des transactions qui appartiennent à X, $\mathcal{J}[j] := \mathcal{T}(X[j])$, $j.time$ est l'estampille temporelle de l'item j, $time(X)$ est l'ensemble des estampilles temporelles de X, $|\mathcal{T}(convoi)|$ est le nombre de transactions associées au $convoi$, $|gPattern|$ et $size(gPattern)$ sont respectivement le nombre de convois et la taille totale des convois dans $gPattern$.

Algorithme 1 : **GeT_Move**

Entrées : ensemble d'occurrences $\mathcal{J}$, int ε, int min_t, ensemble d'items C_{DB}, double θ, int min_c, double min_{wei}

début
  $X := I(\mathcal{T}(\emptyset))$; //La racine
  pour $i := 1$ *to* $|C_{DB}|$ **faire**
    si $|\mathcal{T}(X[i])| \geq \varepsilon$ *et* $|X[i]|$ *est clos* **alors**
      LCM_Iter($X[i], \mathcal{T}(X[i]), i$) ;
fin
LCM_Iter($X, \mathcal{T}(X), i(X)$)
début
  ExtractionMotif(X, min_t) ; /*X est an motif ?*/
  pour tous les *transaction* $t \in \mathcal{T}(X)$ **faire**
    pour tous les $j \in t, j > i(X), j.time \notin time(X)$ **faire**
      insérer j en $\mathcal{J}[j]$;
  pour tous les $j \in \mathcal{J}[j]$ *en order décroissant* **faire**
    si $|\mathcal{T}(\mathcal{J}[j])| \geq \varepsilon$ *et* $\mathcal{J}[j]$ *est clos* **alors**
      LCM_Iter($\mathcal{J}[j], \mathcal{T}(\mathcal{J}[j]), j$) ;
    Effacer $\mathcal{J}[j]$;
fin
ExtractionMotif(X, min_t)
début
  si $|X| \geq min_t$ **alors**
    sortie X ; /*Closed Swarm*/
    $mGroupe := \emptyset; convoi := \emptyset; mc := \emptyset$;
    pour $k := 1$ *to* $|X - 1|$ **faire**
      si $x_k.time = x_{(k+1)}.time - 1$ **alors**
        $convoi := convoi \cup x_k$;
        si $\frac{|\mathcal{T}(x_k) \cap \mathcal{T}(x_{k+1})|}{|\mathcal{T}(x_k) \cup \mathcal{T}(x_{k+1})|} \geq \theta$ **alors**
          $mc := mc \cup x_k$;
        sinon
          si $|mc \cup x_k| \geq min_t$ **alors**
            sortie $mc \cup x_k$; /*Groupes Mobiles*/
          $mc := \emptyset$;
      sinon
        si $|convoi \cup x_k| \geq min_t$ *et* $|\mathcal{T}(convoi \cup x_k)| = |\mathcal{T}(X)|$ **alors**
          sortie $convoi \cup x_k$; /*Convoi*/
          $mGroupe := mGroupe \cup (convoi \cup x_k)$;
        si $|mc \cup x_k| \geq min_t$ **alors**
          sortie $mc \cup x_k$; /*Groupes Mobiles*/
        $convoi := \emptyset; mc := \emptyset$;
    si $|mGroupe| \geq min_c$ *et* $size(mGroupe) \geq min_{wei}$ **alors**
      sortie $mGroupe$; /*Motifs de Groupe*/
fin

Tableau 3. Matrice d'itemsets clos

Block B		b_1	b_2	
Itemsets fréquents et clos CI		ci_{11}	ci_{12}	ci_{22}
O_{DB}	o_1	1		1
	o_2	1		1
	o_3	1	1	
	o_4	1	1	

DEFINITION 8. — *Bloc. Etant donné un ensemble d'estampilles temporelles $T_{DB} = \{t_1, t_2, \ldots, t_n\}$ et une matrice de cluster CM, CM est décomposée verticalement en plus petites matrices équivalentes (en termes d'intervalle), chacune étant considérée comme un bloc b. Nous notons T_b l'ensemble des estampilles associées au bloc b, $T_b = \{t_1, t_2, \ldots, t_k\}$, alors nous avons $|T_b| = k \leq |T_{DB}|$.*

Nous obtenons ainsi un ensemble de blocs $B = \{b_1, b_2, \ldots, b_p\}$ avec $|T_{b_1}| = |T_{b_2}| = \ldots = |T_{b_p}|, \bigcup_{i=1}^{p} b_i = CM$ et $\bigcap_{i=1}^{p} b_i = \emptyset$. Etant donné un ensemble d'ICF $CI = \{CI_1, CI_2, \ldots, CI_p\}$ avec CI_i extrait à partir du bloc b_i. CI est présentée comme une *matrice d'itemsets clos* qui est formée par une connexion horizontale entre tous les ICF locaux : $MIC = \bigcup_{i=1}^{p} CI_i$.

DEFINITION 9. — *Matrice d'itemsets clos (MIC). La matrice d'itemsets clos est une matrice de clusters ayant les spécificités suivantes : 1) L'estampille temporelle t devient un bloc b. 2) L'item c est un ICF ci.*

Par exemple, tableau 3, nous avons deux ensembles de ICF $CI_1 = \{ci_{11}\}, CI_2 = \{ci_{12}, ci_{22}\}$ qui sont respectivement extraits des blocs b_1, b_2. Nous avons MIC qui est obtenue de CI_1, CI_2.

Maintenant, nous appliquons l'algorithme d'extraction de ICF sur la matrice d'itemsets clos MIC, nous retrouvons alors tous les ICF correspondant aux données. Notons que les items (dans MIC) qui sont dans le même bloc ne peuvent pas être dans le même ICF.

LEMME 10. — *Etant donné une matrice de cluster CM, décomposée verticalement en un ensemble de blocs $B = \{b_1, b_2, \ldots, b_p\}$ tels que $\forall \Upsilon, \Upsilon$ est un IFC et Υ est extrait à partir de CM alors Υ peut être extrait de la matrice des itemsets clos MIC.*

PREUVE. Supposons $\forall b_i, \exists I_i$ est un ensemble d'items appartenant à b_i, nous avons $\bigcap_{i=1}^{|B|} I_i = \emptyset$. Si $\forall \Upsilon, \Upsilon$ est un IFC extrait de CM alors Υ est formé de $\Upsilon = \{\gamma_1, \gamma_2, \ldots, \gamma_p\}$ avec γ_i un ensemble d'items tel que $\gamma_i \subseteq I_i$. De plus, Υ est un IFC et $O(\Upsilon) = \bigcap_{i=1}^{p} O(\gamma_i)$ alors $\forall O(\gamma_i), O(\Upsilon) \subseteq O(\gamma_i)$. Nous avons $|O(\Upsilon)| \geq \varepsilon$, donc $|O(\gamma_i)| \geq \varepsilon$ ainsi γ_i est un itemset fréquent. Supposons que $\exists \gamma_i, \gamma_i \notin CI_i$ alors $\exists \Psi, \Psi \in CI_i$ tels que $\gamma_i \subseteq \Psi$ et $\sigma(\gamma_i) = \sigma(\Psi), O(\gamma_i) = O(\Psi)$. Notons que Ψ, γ_i appartiennent à b_i. Comme $O(\Upsilon) = O(\gamma_1) \cap O(\gamma_2) \cap \ldots \cap O(\gamma_i) \cap \ldots \cap O(\gamma_p)$ et que $\exists \Upsilon'$ tel que $O(\Upsilon') = O(\gamma_1) \cap O(\gamma_2) \cap \ldots \cap O(\Psi) \cap \ldots \cap O(\gamma_p)$. Ainsi, $O(\Upsilon') = O(\Upsilon)$ et $\sigma(\Upsilon') = \sigma(\Upsilon)$. Nous savons que $\gamma_i \subseteq \Psi$ et $\Upsilon \subseteq \Upsilon'$. Par consé-

quent, nous obtenons $\Upsilon \subseteq \Upsilon'$ et $\sigma(\Upsilon) = \sigma(\Upsilon')$. Comme Υ n'est pas un IFC, cela contredit l'hypothèse et nous avons si $\exists \gamma_i, \gamma_i \notin CI_i$ alors Υ n'est pas un itemset fréquent clos. Finalement, nous pouvons conclure que $\forall \Upsilon, \Upsilon = \{\gamma_1, \gamma_2, \ldots, \gamma_p\}$ est un IFC extrait de CM, $\forall \gamma_i \in \Upsilon$, γ_i doit appartenir à CI_i et γ_i est un item dans la matrice des items clos MIC. Donc, nous pouvons affirmer que Υ peut être extrait en appliquant un algorithme d'extraction d'IFC sur MIC.

En appliquant le *lemme 10*, nous pouvons extraire tous les IFC et à partir des itemsets, nous pouvons extraire les motifs spatiotemporels. Notons que Incremental GeT_Move ne dépend pas de la valeur minimale de min_t. La raison en est la suivante : min_t est uniquement utilisée dans l'étape d'extraction des motifs et non celle des itemsets. Quel que soit min_t ($min_t \geq$ taille des blocs ou $min_t \leq$ taille des blocs), Incremental GeT_Move peut extraire tous les IFC et donc le résultat obtenu est le même.

L'algorithme *Incremental GeT_Move* est détaillé dans l'algorithme 2. La principale différence avec le contenu de *GeT_Move* est la sous-fonction *Update* pour laquelle, nous générons étape par étape la matrice des itemsets clos à partir des blocs ((ligne 14 et lignes 22-26). Ensuite, nous appliquons *GeT_Move* pour extraire les motifs (ligne 5).

5. Expérimentations

Les performances globales ont été évaluées sur des données réelles et des données synthétiques. Tous les algorithmes sont implémentés en C++, et toutes les expériences sont réalisées sur un système 2,8 GHz Intel Core i7 avec 4 Go de mémoire. Le système fonctionne sous Ubuntu 11.10 et g++ version 4.6.1.

Le code source est intégré dans notre système de démonstration en ligne [4]. Comme dans (Li, Ding *et al.*, 2010), nous ne présentons que les résultats sur les données suivantes [5] : *Swainsoni dataset* 43 objets évoluant au cours de 764 dates. Le lecteur intéressé peut se référer à notre système de démonstration en ligne[2] pour les autres résultats expérimentaux. En outre, de même que (Li, Ding *et al.*, 2010) (Jeung, Yiu *et al.*, 2008) (Jeung, Shen, Zhou, 2008), nous utilisons d'abord une interpolation linéaire pour combler les données manquantes, puis DBScan (Ester *et al.*, 1996) ($MinPts = 2, Eps = 0.001$) est appliqué pour générer les clusters à chaque estampille temporelle.

Pour réaliser les différentes comparaisons, nous utilisons $CMC, CuTS^*$ [6] (*convoy mining*) et $ObjectGrowth$ (*closed swarm mining*). Notons que dans (Li, Ding *et al.*, 2010), $ObjectGrowth$ produit de meilleures performances que $VG - Growth$ (Wang *et al.*, 2006) (un algorithme d'extraction de motif de groupe). C'est pourquoi

4. www.lirmm.fr/∼phan/index.jsp
5. http ://www.movebank.org
6. Le code source de $CMC, CuTS^*$ est disponible à http ://lsirpeople.epfl.ch/jeung/source_codes.htm

Algorithme 2 : **Incrémental GeT_Move**

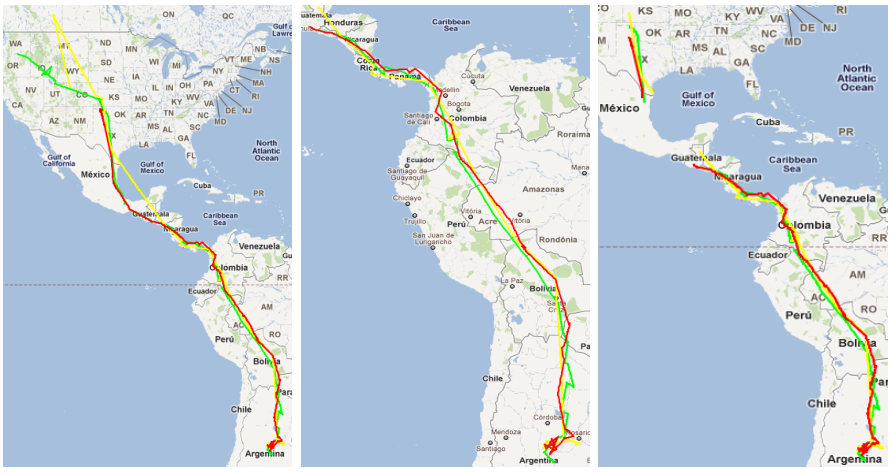
Entrées : ensemble d'occurrence K, int ε, int min_t, double θ, blocs B, int min_c, double min_{wei}

début
  $K := \emptyset; CI := \phi$; int $item\_total := 0$;
  pour tous les $b \in B$ **faire**
    LCM(b, ε, I_b) ;
  GeT_Move($K, \varepsilon, min_t, CI, \theta, min_c, min_{wei}$) ;
fin
LCM(ensemble d'occurrence $\mathcal{J}$, int σ_0, ensemble d'items C)
début
  $X := I(\mathcal{T}(\emptyset))$; //La racine
  pour $i := 1$ *to* $|C|$ **faire**
    si $|\mathcal{T}(X[i])| \geq \varepsilon$ *et* $|X[i]|$ *est clos* **alors**
      LCM_Iter($X[i], \mathcal{T}(X[i]), i$) ;
fin
LCM_Iter($X, \mathcal{T}(X), i(X)$)
début
  Réactualiser($K, X, \mathcal{T}(X), item\_total$**++**) ;
  pour tous les *transaction* $t \in \mathcal{T}(X)$ **faire**
    pour tous les $j \in t, j > i(X), j.time \notin time(X)$ **faire**
      insérer j en $\mathcal{J}[j]$;
  pour tous les $j, \mathcal{J}[j] \neq \phi$ *en order décroissant* **faire**
    si $|\mathcal{T}(\mathcal{J}[j])| \geq \varepsilon$ *et* $\mathcal{J}[j]$ *est clos* **alors**
      LCM_Iter($\mathcal{J}[j], \mathcal{T}(\mathcal{J}[j]), j$) ;
    Effacer $\mathcal{J}[j]$;
fin
Réactualiser($K, X, \mathcal{T}(X), item\_total$)
début
  pour tous les $t \in \mathcal{T}(X)$ **faire**
    insérer $item\_total$ en $K[t]$;
  $CI := CI \cup item\_total$;
fin

nous ne considérons que la version $ObjectGrowth$ et non les deux. Dans les résultats, GeT_Move et Incremental GeT_Move extraient des swarms clos, des convois, des motifs de groupe alors que $CMC, CuTS^*$ extraient uniquement des convois et $ObjectGrowth$ extrait uniquement des swarms clos.

5.1. Quels motifs sont extraits ?

Nous avons montré que l'extraction de motifs spatiotemporels peut être associée à la problématique d'extraction d'itemsets fréquents. De ce fait, d'un point de vue théorique, notre approche peut fournir de bons résultats. De façon expérimentale, nous réalisons une comparaison des motifs obtenus, tout d'abord avec $CMC, CuTS^*$,



(a) (b) (c)

Figure 8. Motifs extraits de Swainsoni (a) swarms clos (b) convois (c) motifs de groupe

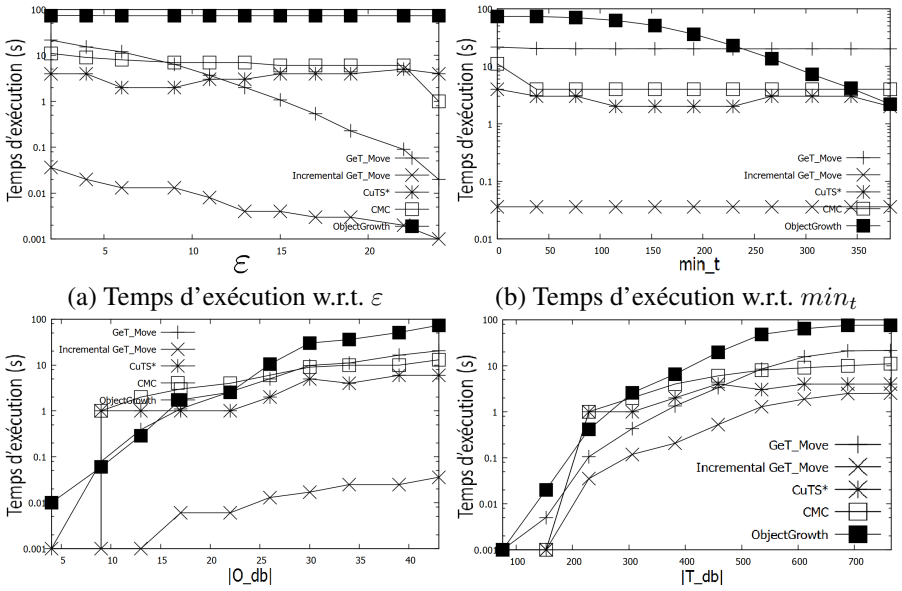
$ObjectGrowth$ puis avec notre approche. Pour appliquer nos algorithmes, nous avons décomposé la matrice de clusters en blocs, chaque bloc b contenant 25 dates. De plus, pour retrouver tous les motifs spatiotemporels, la valeur par défaut dans les tests réalisés est de 2 pour ε (deux objets peuvent former un motif), min_t est valué à 1. Il faut souligner que ces valeurs par défaut sont les moins indulgentes pour les algorithmes. C'est pourquoi dans la suite, nous faisons varier les valeurs de min_t afin d'obtenir d'autres convois, de swarms clos et de motifs de groupe. Pour ces derniers, min_c est égal à 1 et min_t à 0.

Les résultats obtenus soulignent que notre approche fournit les mêmes résultats que les algorithmes de la littérature. Pour chacun des types de motifs extraits, un exemple est proposé figure 8.

5.2. Evaluation des performances

Pour évaluer les performances de nos algorithmes, nous avons généré des données d'objets mobiles synthétiques en grand volume à l'aide du générateur de Brinkhoff [7] comme dans (Li, Ding *et al.*, 2010). Nous avons généré 500 objets ($|O_{DB}| = 500$) pour 10^4 estampilles temporelles ($|T_{DB}| = 10^4$) avec des vitesses d'évolution lentes

7. http ://iapg.jade-hs.de/personen/brinkhoff/generator/



(a) Temps d'exécution w.r.t. ε

(b) Temps d'exécution w.r.t. min_t

(c) Temps d'exécution w.r.t. $|O_{DB}|$

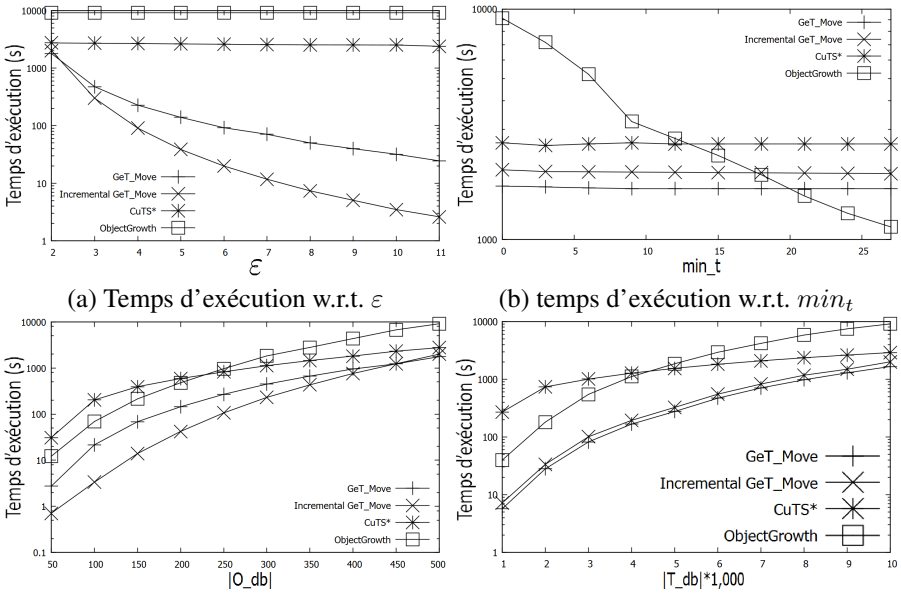
(d) Temps d'exécution w.r.t. $|T_{DB}|$

Figure 9. Temps d'exécution - Jeu de données Swainsoni

dans l'espace. Il y a 5×10^6 points au total. DBScan ($MinPts = 3, Eps = 300$) est appliqué pour obtenir les clusters de chaque estampille temporelle.

Efficacité w.r.t. ε, min_t. Les figures 9a et 10a indiquent les temps d'exécution w.r.t. ε. Il apparaît que nos deux algorithmes sont nettement plus efficaces que les autres. ObjectGrowth obtient les moins bonnes performances, la raison principale est associée à un faible min_t (par défaut $min_t = 1$), la règle d'élagage de *Apriori* ne peut être réalisée. Ainsi, l'espace de recherche peut être parcouru dans sa globalité ($2^{|O_{DB}|}$ dans le pire des cas). Il n'y a pas non plus de règle d'élagage pour ε et les variations de ε n'affectent donc pas les temps d'exécution d'ObjectGrowth. GeT_Move est plus lent qu'Incremental GeT_Move, la raison principale en est la suivante : GeT_Move doit gérer un plus grand nombre d'items et de plus longs items. Grâce au principe des blocs, le nombre et la taille des itemsets sont grandement réduits.

Les figures 9b et 10b indiquent les temps d'exécution w.r.t. min_t. Dans la plupart des cas, les performances de nos approches sont meilleures (voir la figure 10b) en particulier avec un min_t de faible valeur. Quand min_t est plus élevé ($min_t > 20$ sur la figure 10b), nos propositions nécessitent un temps d'exécution plus long que CuTS* et ObjectGrowth. Dans ce cas, le nombre de motifs est considérablement réduit (Sur les figures 11b, 12b) (c'est-à-dire qu'il n'y a aucun convoi avec $min_t > 100$ (resp. $min_t > 10$), figure 11b (resp. figure 12b)) et donc CuTS* et ObjectGrowth se



(a) Temps d'exécution w.r.t. ε (b) temps d'exécution w.r.t. min_t

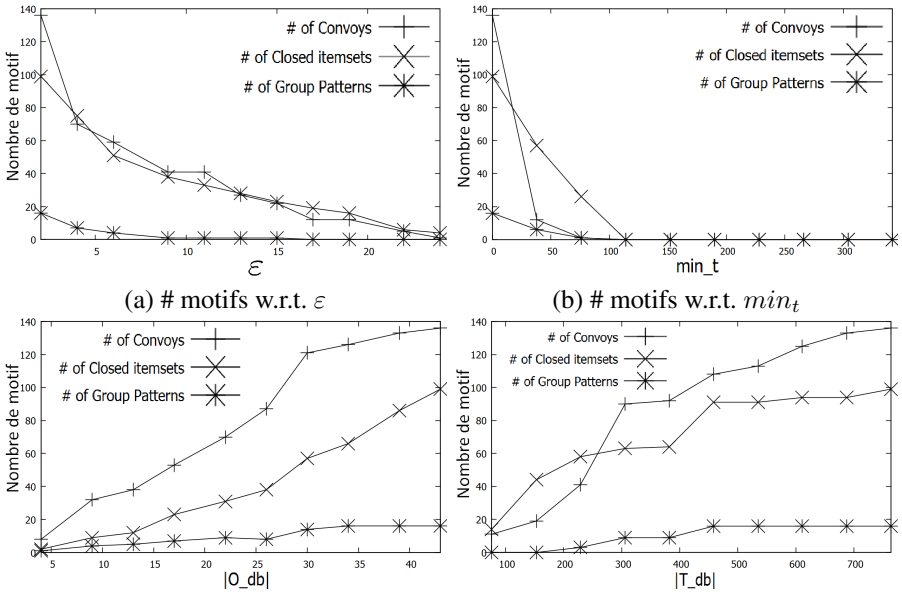
(c) Temps d'exécution w.r.t. $|O_{DB}|$ (d) temps d'exécution w.r.t. $|T_{DB}|$

Figure 10. Temps d'exécution - Jeu de données synthétiques

terminent rapidement. Alors que GeT_Move et Incremental GeT_Move génèrent des itemsets fréquents.

Efficacité w.r.t. $|O_{DB}|, |T_{DB}|$. Les figures 9c-d et 10c-d indiquent les temps d'exécution lorsque $|O_{DB}|$ et $|T_{DB}|$ varient. Dans tous les cas, Incremental GeT_Move obtient les meilleures performances. Alors qu'avec le jeu de données synthétiques, (figure 10d) et des valeurs plus faibles de $\varepsilon = 2$ et $min_t = 1$, GeT_Move est légèrement plus rapide.

Passage à l'échelle w.r.t. ε. Nous pouvons noter que les temps d'exécution des algorithmes ne changent pas de façon significative lorsque $min_t, |O_{DB}|, |T_{DB}|$ varient pour le jeu de données synthétiques (figure 10). Néanmoins, ils sont sensibles aux variations de ε (par défaut $min_t = 1$). C'est pourquoi nous avons généré un autre jeu de données synthétiques afin de tester le passage à l'échelle des algorithmes selon ε. Ce jeu de données contient 50 000 objets avec 10 000 estampilles temporelles et 500 millions de points au total. L'exécution de CMC et CuTS* échoue du fait d'un problème d'allocation mémoire après avoir géré 300 millions de points. ObjectGrowth, quant à lui, ne fournit pas de résultat après un jour entier d'exécution. La raison principale est la valeur basse de min_t $(= 1)$ qui conduit à un espace de recherche important $(\approx 2^{50000})$. Nos algorithmes, grâce à LCM, fournissent des résultats en quelques heures (figure 13a).



(a) # motifs w.r.t. ε (b) # motifs w.r.t. min_t

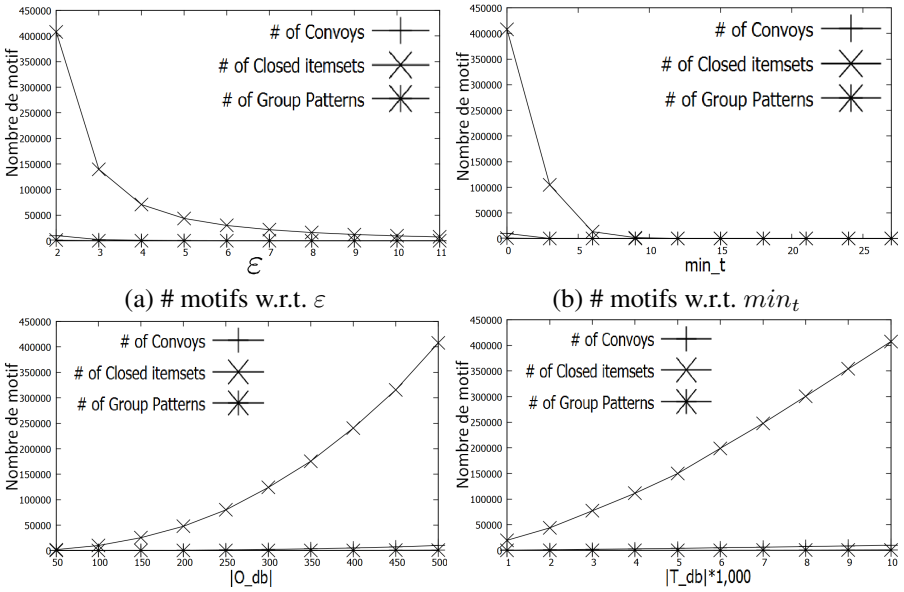
(c) # motifs w.r.t. $|O_{DB}|$ (d) # motifs w.r.t. $|T_{DB}|$

Figure 11. # motifs - Jeu de données Swainsoni # IFC est égal # swarms clos

Efficacité w.r.t. *Block-size*. Pour évaluer la meilleure taille des blocs, nous avons exécuté Incremental GeT_Move en utilisant les valeurs par défaut de ϵ, min_t avec différentes tailles de bloc sur les jeux de données (note : Buffalo[3],$|O_{DB}| = 165, |T_{DB}| = 3,000$) et ($|O_{DB}| = 500, |T_{DB}| = 1,000$). La taille des blocs pour laquelle Incremental GeT_Move obtient les meilleures performances est située entre 20 et 30 estampilles temporelles (figure 13b). L'explication est liée au fait que les objets évoluent ensemble pendant un court intervalle (de 20 à 30 estampilles environ). Donc, en positionnant la taille des blocs de cette façon, les données sont efficacement compressées dans les IFC. Alors qu'avec une taille de bloc supérieure, les mouvements des objets ne sont pas résumés de façon aussi efficace. Pour une taille entre 5 et 15, nous devons gérer de nombreux blocs et donc le processus est ralenti. Dans les expériences précédentes, la taille des blocs était fixée à 25.

6. Conclusion

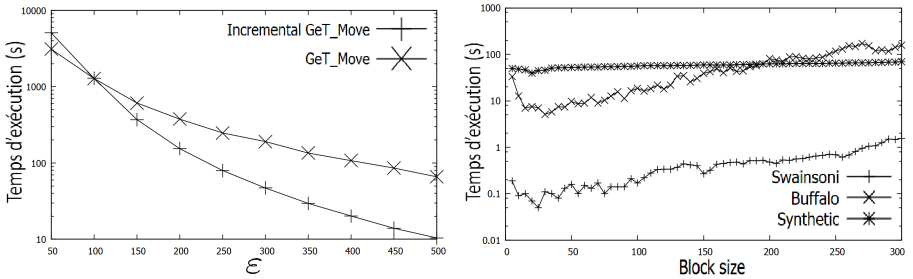
Dans cet article, nous avons proposé une approche unifiée d'extraction de motifs spatiotemporels en appliquant des algorithmes de recherche d'itemsets fréquents clos. Les expérimentations conduites à la fois sur des données réelles et des données synthétiques ont souligné l'efficacité de la proposition. Dans les perspectives associées à ces travaux, nous souhaitons prendre en compte l'arrivée de nouveaux objets qui



(a) # motifs w.r.t. ε (b) # motifs w.r.t. min_t

(c) # motifs w.r.t. $|O_{DB}|$ (d) # motifs w.r.t. $|T_{DB}|$

Figure 12. # motifs - Jeu de données synthétiques # IFC est égal # swarms clos



(a) Temps d'exécution w.r.t ε (b) Temps d'exécution w.r.t taille des blocs

Figure 13. (a) Temps d'exécution w.r.t ε - Jeu de données synthétiques (b) temps d'exécution w.r.t taille des blocs

n'étaient pas présents lors de la première extraction sachant que nous sommes en mesure de traiter de nouvelles données des objets déjà existants. Nous avons, en effet, fait l'hypothèse que les objets sont toujours les mêmes ce qui n'est pas toujours le cas dans les applications réelles de trajectoire d'objets mobiles.

Références

Agrawal R., Srikant R. (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th international conference on very large data bases*, p. 487–499.

Bogorny V., Shekhar S. (2010). Spatial and spatio-temporal data mining. In *Icdm*, p. 1217.

Cao H., Mamoulis N., Cheung D. W. (2006). Discovery of collocation episodes in spatiotemporal data. In *Proceedings of the sixth international conference on data mining*, p. 823–827.

Ester M., Kriegel H.-P., Sander J., Xu X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd'96*, p. 226-231.

Gudmundsson J., Kreveld M. van. (2006). Computing longest duration flocks in trajectory data. In *Proceedings of the 14th annual acm international symposium on advances in geographic information systems*, p. 35–42. New York, NY, USA, ACM. http://doi.acm.org/10.1145/1183471.1183479

Hai P. N., Ienco D., Poncelet P., Teisseire M. (2012a). Extracting trajectories through an efficient and unifying spatio-temporal pattern mining system. In *Ecml/pkdd (2)*, p. 820-823.

Hai P. N., Ienco D., Poncelet P., Teisseire M. (2012b). Mining fuzzy moving object clusters. In *Adma*, p. 100-114.

Hai P. N., Ienco D., Poncelet P., Teisseire M. (2012c). Mining time relaxed gradual moving object clusters. In *Acm sigspatial gis*, p. 478-481.

Han J., Li Z., Tang L. (2010). Mining moving object, trajectory and traffic data. In H. Kitagawa, Y. Ishikawa, Q. Li, C. Watanabe (Eds.), *Database systems for advanced applications*, vol. 5982, p. 485-486. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-12098-5_56

Han J., Pei J., Yin Y. (2000, mai). Mining frequent patterns without candidate generation. *SIGMOD Rec.*, vol. 29, n° 2, p. 1–12.

Jensen C., Lin D., Ooi B. C. (2007, sept.). Continuous clustering of moving objects. *Knowledge and Data Engineering, IEEE Transactions on*, vol. 19, n° 9, p. 1161 -1174.

Jeung H., Shen H. T., Zhou X. (2008). Convoy queries in spatio-temporal databases. In *Proceedings of the 2008 ieee 24th international conference on data engineering*, p. 1457–1459.

Jeung H., Yiu M. L., Zhou X., Jensen C. S., Shen H. T. (2008, août). Discovery of convoys in trajectory databases. *Proc. VLDB Endow.*, vol. 1, n° 1, p. 1068–1080. http://dl.acm.org/citation.cfm?id=1453856.1453971

Kalnis P., Mamoulis N., Bakiras S. (2005). On discovering moving clusters in spatio-temporal data. In *Proceedings of the 9th international conference on advances in spatial and temporal databases*, p. 364–381. Berlin, Heidelberg, Springer-Verlag. http://dx.doi.org/10.1007/11535331_21

Lee J.-G., Han J., Whang K.-Y. (2007). Trajectory clustering : a partition-and-group framework. In *Proceedings of the 2007 acm sigmod international conference on management of data*, p. 593–604.

Li Z., Ding B., Han J., Kays R. (2010, September). Swarm : mining relaxed temporal moving object clusters. *Proc. VLDB Endow.*, vol. 3, nº 1-2, p. 723–734. http://dl.acm.org/citation .cfm?id=1920841.1920934

Li Z., Ji M., Lee J.-G., Tang L.-A., Yu Y., Han J. *et al. (2010). Movemine : mining moving object databases. In* Proceedings of the 2010 acm sigmod international conference on management of data*, p. 1203–1206.*

Mamoulis N., Cao H., Kollios G., Hadjieleftheriou M., Tao Y., Cheung D. W. (2004). Mining, indexing, and querying historical spatiotemporal data. In Proceedings of the tenth acm sigkdd international conference on knowledge discovery and data mining*, p. 236–245.*

Romero A. (2011). Mining moving flock patterns in large spatio-temporal datasets using a frequent pattern mining approach. In Master thesis, university of twente, faculty itc.

Tang L. A., Zheng Y., Yuan J., Han J., Leung A., Hung C.-C. et al. (2012). On discovery of traveling companions from streaming trajectories. In *Icde*, p. 186-197.

Uno T., Kiyomi M., Arimura H. (2004). Lcm ver. 2 : Efficient mining algorithms for frequent/closed/maximal itemsets. In *Fimi.*

Verhein F. (2009). Mining complex spatio-temporal sequence patterns. In *In proceedings of the ninth siam international conference on data mining.*

Vieira M. R., Bakalov P., Tsotras V. J. (2009). On-line discovery of flock patterns in spatio-temporal data. In *Proceedings of the 17th acm sigspatial international conference on advances in geographic information systems*, p. 286–295. New York, NY, USA, ACM. http://doi.acm.org/10.1145/1653771.1653812

Wang Y., Lim E.-P., Hwang S.-Y. (2006, jun). Efficient mining of group patterns from user movement data. *Data Knowl. Eng.*, vol. 57, nº 3, p. 240–282.

Zheng K., Zheng Y., Yuan J., Shang S. (2013). On discovery of gathering patterns from trajectories. In *Icde.*