# Chapitre 2 : Contribution de la fouille de données à l'analyse de comportements

## 2.1. Introduction

La communauté des chercheurs en fouille de données (*data mining)* est très active. Plusieurs travaux de fouille de données, fouille de données spatiales et/ou temporelles (Roddick et al., 2001) ont vu le jour dans des domaines d'application très variés comme la gestion de la relation client (identification de prospects, churn[39], etc.), le marketing stratégique (mailing[40], association de produits, etc.), la gestion des risques (remboursement de crédits, détection de fraudes, etc.) et la prévision du trafic routier. L'acquisition de connaissances par induction a donc démontré son succès par son large panel d'applications. La fouille de données est passée rapidement du cadre de la recherche vers l'industrie pour occuper une place importante dans le domaine de l'aide à la décision (Tufféry 2010)[41].

La fouille de données quel que soit son domaine, classique, spatial, d'objets mobiles ou du trafic peut contribuer à l'analyse des comportements d'objets en déplacements. Les situations et les mouvements intéressants de mobiles peuvent être mis en exergue en détectant des relations et des motifs cachés décrivant des comportements.

Ce chapitre est organisé en trois parties : la première partie décrit les domaines de la fouille de données à savoir la fouille de données classique, la fouille de données spatiales, la fouille de données d'objets mobiles et la fouille de données de trafic d'objets. La deuxième partie, expose deux prototypes d'analyse de comportements basés sur la fouille de données. Enfin la troisième partie, présente une synthèse des méthodes et algorithmes de fouilles de données.

## 2.2. Les domaines de la fouille de données

Les volumes importants de données générés actuellement, offrent un potentiel important à la fouille de données. Les bases de données peuvent contenir de nombreuses informations d'intérêt, cachées par le volume important de données. Il est naturel de vouloir exploiter ces données et extraire ces informations. Cependant, le volume de données et leur exploitation n'évoluent pas de la même manière. Les données enregistrent une croissance exponentielle alors que leur exploitation est linéaire, ce qui engendre des

[39] Identification de clients qui sont susceptible de partir à la concurrence.
[40] Le publipostage est l'adaptation de la communication aux différents segments de clientèle.
[41] Page 13.

pertes d'informations non-négligeables. Avec les méthodes d'analyse statistiques, il est difficile de traiter des millions d'enregistrements, avec plusieurs centaines de variables ayant des types de plus en plus complexes (texte, géométrie, réseau, etc.). Ces méthodes atteignent leurs limites face aux volumes de données de plus en plus importants. Pour faire face à ce problème, la fouille de données a été inventée. Les enjeux actuels liés à la concurrence du marché, le besoin de rapidité de traitement des données, de prise de décision ont poussé les industries à s'approprier la fouille de données rapidement (Tufféry 2007)[42].

Quant à ce qui l'a fait émerger, il s'agit du développement des moyens de stockage et de calcul informatique, l'évolution du domaine décisionnel et la possibilité de prendre en compte les bruits présents dans les données collectées (données lacunaires). La fouille de données a rapidement été utilisée dans plusieurs domaines qui partent de l'infiniment petit comme la génomique à l'infiniment grand comme l'astrophysique (Tufféry 2007)[43]. Etant donné les potentialités offertes par la fouille de données, son spectre d'application est très large. Elle est utilisée en commerce pour l'analyse des comportements de consommation par exemple, dans les banques pour la distinction entre les bons et mauvais payeurs et en assurance pour l'identification des fraudes et les critères explicatifs des risques.

Selon la nature des objets étudiés, plusieurs domaines de fouille de données peuvent intervenir. Nous présentons ci-après un panorama des méthodes de fouille de données organisées par domaine, à savoir la fouille de données classique, spatiales, d'objets mobiles et du trafic. La fouille de données classique s'intéresse à l'exploration de données relationnelles, la fouille de données spatiales s'intéresse à l'exploration de données spatiales et la fouille de données d'objets mobiles et de trafic de mobiles sont des domaines plus récents qui s'intéressent à l'exploration de données de capteurs de mobiles.

[42] Page 13
[43] Page 1

Aujourd'hui, énormément de problèmes, de méthodes et d'algorithmes de fouille de données existent dans la littérature. Chaque domaine possède une panoplie de problèmes d'extraction de connaissances, chaque problème détient plusieurs méthodes et chaque méthode dispose de plusieurs algorithmes comme illustré à la figure 2-1.

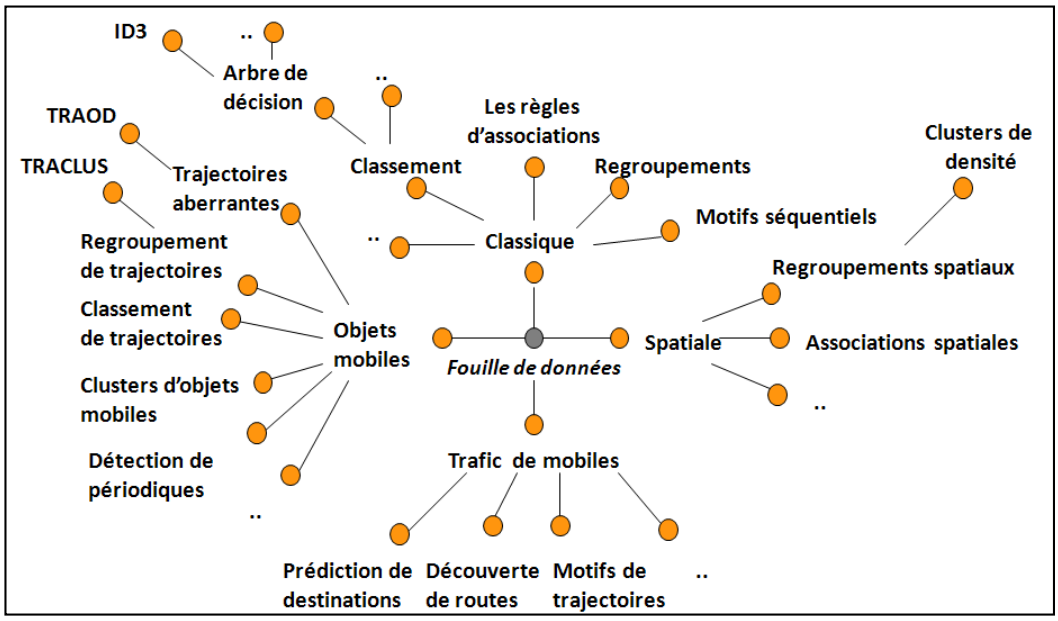



Figure 2-1 : Organisation de l'état de l'art de la fouille de données.

Nous allons présenter dans les sous sections suivantes, un état de l'art sur la fouille de données. Dans cet état de l'art, nous détaillerons par domaine les méthodes et algorithmes nécessaires pour la compréhension de ce travail.

## 2.2.1. La fouille de données classique

Nous appelons fouille de données classique, l'ensemble des problèmes de fouille de données qui ne considèrent pas les relations spatiales. Les objets étudiés sont souvent des tuples de bases de données ne contenant pas de dimension spatiale (localisation absolue). Les tuples peuvent être définis comme des ensembles de valeurs d'attributs ordonnées relatifs à un enregistrement (observation). Ces tuples sont organisés sous forme de structures de données en table[44] qui sont reliées par des relations logiques. Pour ce qui est des relations spatiales, elles vont être définies dans la section 2.2.2.

[44] Une table en base de données est une structure en tableau où les lignes sont des enregistrements et les colonnes sont des attributs

Nous allons exposer dans les sous-sections suivantes, les différents problèmes de la fouille de données classique : les associations, le classement, le groupement, la fouille des séries chronologiques et l'analyse des aberrations.

### 2.2.1.1. Les associations

Les associations sont des problèmes non supervisés de fouille de données permettant d'extraire des relations d'implication au sein d'un même événement ou entre des séquences d'événements ordonnés dans le temps. Nous allons présenter ci-après, les deux problèmes d'association qui existent, à savoir, la recherche de règles d'association et la recherche de motifs séquentiels.

#### *2.2.1.1.1. Les Règles d'association*

Les règles d'association sont des règles d'occurrence mettant en exergue les relations cachées par le volume important des bases de données. Elles sont basées sur la découverte de motifs fréquents qui sont des ensembles d'items apparaissant souvent ensemble dans les bases de données. Un item étant une combinaison entre un attribut et une valeur. Formellement, une base de données D est définie par un triplet (O, P, R) tel que O est l'ensemble fini des objets de la base (appelé individus), P l'ensemble fini d'attributs (appelés variables) et R une relation binaire entre les deux ensembles O et P. Etant p un itemset appartenant à P, on dit que o contient p si (o,p) appartient à R, tel que o appartient à O. La découverte de règles d'association consiste à sélectionner tous les ensembles d'items I inclus dans P qui sont fréquents dans O. Ces ensembles d'items appelés itemsets sont utilisés pour générer les règles d'association r de la forme $\boldsymbol{A \rightarrow B}$ tel que : ***A*** et ***B*** sont deux itemsets fréquents et leur intersection est vide. ***A*** est appelé antécédent de la règle ou partie gauche et ***B*** est appelé conséquent ou partie droite. Dans un processus d'extraction de règles d'association, seuls les itemsets ayant un bon support et les règles ayant une bonne confiance sont conservées. Le support est un indicateur de fiabilité de la règle, appelé aussi porté. Il est égal à la fréquence d'apparition d'un itemset dans les transactions par rapport au nombre total de transactions de la base de données (Tufféry 2007). La mesure support est évaluée comme suit :

***Supp(A → B)= card(AB)/card (O),*** *card* est le cardinal de l'ensemble ou la fréquence d'apparition.

La confiance représente l'indicateur de précision de la règle qui est égal à la fréquence d'un itemset par rapport à la fréquence d'apparition de la première partie de la règle d'implication (Tufféry 2007). La mesure confiance est évaluée comme suit :

***Conf(A → B)= card(AB)/card (A).***

Le principe est le suivant, les supports et les confiances devraient être calculés pour toutes les règles possibles puis comparés aux seuils "minsupp" et "minconf" qui sont définis *a priori* par les utilisateurs. Les règles ayant les valeurs des paramètres support et confiance au-delà des seuils donnés par les utilisateurs sont conservées car elles possèdent des relations dites fortes. Le coût important de la recherche de toutes les règles a poussé à découpler les conditions de support et de confiance[45]. Le problème est traité le plus souvent en deux phases :

1. Génération d'itemsets satisfaisant les conditions de support minimal,
2. Génération des règles satisfaisant les conditions de confiance à partir d'itemsets précédemment générés.

Il existe d'autres mesures proposées dans la littérature pour améliorer la sélectivité car trop de règles sont extraites dont certaines sans intérêt. Parmi ces mesures, nous pouvons citer les mesures Lift, Pearl, Loevinger, Surprise et J-mesure. Nous ne pouvons pas exposer toutes les mesures dans ce manuscrit, mais nous renvoyons pour plus d'informations, aux travaux de S. Lallich (Vaillant et al., 2005) (Le Bras et al., 2011) qui traite en détail cette question.

Une mesure de performance souvent utilisée avec le support et la confiance pour évaluer l'amélioration apportée par une règle par rapport au hasard est la mesure Lift. Le résultat de cette mesure permet d'identifier la corrélation entre les deux parties de la règle d'association :

- Un Lift supérieur à 1 indique une corrélation positive,
- Un Lift égal à 1 indique une corrélation nulle,
- Un Lift inférieur à 1 indique une corrélation négative.

La mesure Lift est évaluée comme suite :

[45] Valérie Monbet, Université Rennes 1,
Lien http://perso.univ-rennes1.fr/valerie.monbet/doc/cours/IntroDM/Chapitre5.pdf

***Lift(A → B)= Conf(A → B)/card (B).***

L'exemple du panier de la ménagère (Agrawal et al., 1993) illustre bien la recherche de règles d'association. Une base de données de transactions (les paniers) est composée d'items (les produits achetés). La découverte des associations consiste à chercher les itemsets (ensemble d'items) fréquemment liés dans une même transaction. Les règles extraites sont par exemple « {bière}→{couches} (30%, 80%) : 80% des gens qui achètent de la bière, achètent également des couches et ces clients représentent 30% des cas». Cette règle est entre items singletons et les règles entre plusieurs items peuvent être plus susceptibles d'impressionner car elles sont plus difficiles à identifier. En sachant quels items sont fréquemment achetés ensemble, une stratégie marketing peut être développée. La découverte de relations entre les items permet par exemple d'organiser les rayons en mettant les produits fréquemment achetés ensemble à proximité pour augmenter les ventes.

Il y a toute une batterie d'algorithmes d'extraction de règles d'association qui existent comme Apriori, FP-growth, TreeProjection. Ces algorithmes différents du point de vue des performances (temps et espace requis d'exécution). FP-growth (Han et al., 2000) par exemple améliore la méthode Apriori en utilisant une représentation des itemsets sous forme d'index appelé FP-Tree (Frequent Pattern Tree) pour éviter de scanner la base de données plusieurs fois. Gardarin (Gardarin et al., 1998) propose aussi une amélioration basée sur un index bitmap pour accélérer le calcul des supports.

Apriori (Finding Frequent Itemsets Using Candidate Generation) (Agrawal & Srikant 1994) est l'algorithme fondateur de l'extraction de règles d'association. Cet algorithme commence par une phase de recherche des itemsets fréquents (candidats) en balayant la base de données. Une structure en treillis permet de faire une génération des itemsets par niveau (itemsets de longueurs *1, 2,..,k*, *k* étant la longueur maximale des itemsets de la base de données). Un itemset est considéré comme fréquent si le support calculé est supérieur ou égal au support minimal fixé par l'utilisateur (condition de support). Dans la deuxième phase, Apriori sauvegarde les règles de type $A \rightarrow B$ dont la confiance dépasse la confiance minimale fixée par l'utilisateur (condition de confiance). A et B sont des itemsets fréquent et $A \cap B = \{\}$. Pour plus de détail, voir l'algorithme *Apriori* (Algorithme 2-1).

**Algorithme Apriori**

**Entrée** : minsupp, minconf, D (base de données)
**Sortie** : L (ensembles d'items fréquents), R (règles d'association)

**Début**
**Phase I** :
K=1 ; L=∅ ; C1= {Candidats de taille 1} ; //initialisation
L1=Frequent(1,C1) ; L=L∪L1 ; //garder les items fréquents
**Tant que** Lk=∅
K++ ;
Ck = Candidats(K,Lk-1); //générer les itemsets de taille K
Lk = Frequent(K,Ck) ; L=L∪Lk ;//garder les items fréquents
**Fin Tant que**;

**Phase II** :
R=∅;
Pour chaque ensemble I de L
Pour chaque sous-ensemble S non vide de I
**Si** Conf (S → I-S) >= minconf
r= " S → ( I-S ) " ;
R=R ∪ {r} ;
**Fin Si**

**Fin**

Algorithme 2-1 : Algorithme Apriori.

Pour des raisons d'optimisation et de réduction de l'espace de recherche, les données sont organisées dans un treillis de Galois. En effet, cette structure de données permet d'ordonner l'ensemble des itemsets selon la relation d'inclusion ensembliste et d'exploiter la fermeture du support (Antimonotonie). L'antimonotonie de la condition de support est une propriété qui va aider à l'élagage du treillis en supprimant les branches qui n'ont pas le support, c'est à dire les itemsets qui ont un support s% inférieur au seuil "minsupp" défini *a priori* par les utilisateurs (voir section 2.2.1.1.1). En effet, si le support d'un ensemble est de s% alors le support de tout sous-ensemble de cet ensemble est au moins égal à s%. La contre-apposée donne, si un ensemble n'a pas le support alors tous les ensembles supérieurs contenant cet ensemble ne l'auront pas aussi. Ce résultat va réduire massivement l'ensemble des itemsets en élaguant toutes les branches supérieures du treillis qui sont reliées aux ensembles n'ayant pas le support.

Nous présentons sur la Figure 2-2 un treillis de Gallois ordonné par niveau où chaque niveau k compte tous les ensembles formés par la combinaison de k itemsets (itemset de longueur k). Il est aisément remarquable que le nombre d'itemsets dans le treillis augmente d'une manière combinatoire.

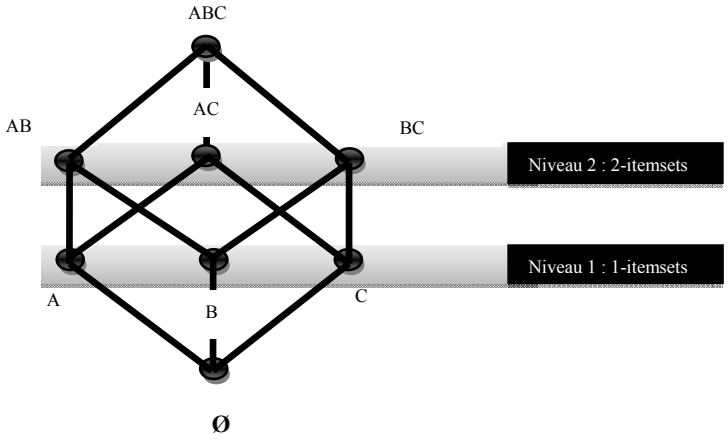



Figure 2-2 : Treillis de Galois.

### *2.2.1.1.2. Les Motifs séquentiels*

Le problème de fouille de motifs séquentiels a été introduit pour la première fois par Srikant et Agrawal (Srikant & Agrawal 1996). La recherche de motifs séquentiels peut être vue comme une extension de la notion de règles d'association, intégrant des contraintes temporelles. Cette recherche met en évidence des associations entre les transactions alors que la recherche des règles d'association détermine les liens au sein d'une même transaction. Les motifs séquentiels sont extraits à partir de séquences d'événements ordonnées et souvent sauvegardés dans des bases de données transactionnelles (voir l'exemple de la Table 2-1). Ces événements ne contiennent pas forcement la notion de temps d'une manière concrète mais elle est implicite dans l'ordre des événements (Han & Kamber 2006)[46]. Une règle séquentielle dont la notion de temps est implicite peut être « 60% des navires qui prennent un tronçon de chemin A, arrivent au point B ». Si la temporalité est présente d'une manière explicite dans les transactions et l'algorithme utilisé intègre les contraintes temporelles, la règle peut devenir «60% des navires qui prennent un tronçon de chemin A, arrivent au point B **dans les 2 heures qui suivent** ».

[46] Page 498

| Identifiant_séquence | Séquence |
|---|---|
| 1 | <*a(**ab**c)(a**c**)d(cf)*> |
| 2 | <*(ad)c(bc)(ae)*> |
| 3 | <*(ef)(**ab**)(df)**c**b*> |
| 4 | <*eg(af)cbc*> |

Table 2-1 : Exemple de base de données séquentielles. La base est au format horizontal, elle contient 4 séquences d'événements, la 1ère séquence contient 5 évènements et le 2ème événement de cette séquence a un itemset (ab)c de taille 3. La recherche de la fréquence d'apparition de cette séquence dans la base donne 2 (itemset en rouge gras) ((Han & Kamber 2006), page 499).

Dans certains contextes, l'identification des événements d'individus au cours du temps est indispensable afin de pouvoir suivre leurs comportements séquentiels (Srikant & Agrawal 1996). Plusieurs applications utilisent ce type de problème pour l'extraction des comportements séquentiels comme l'analyse de séquences biologiques, l'analyse des séquences naturelles et l'analyse des comportements des utilisateurs qui naviguent sur le web (enchaînement des clics de souris sur une page web, etc.)

Plusieurs algorithmes d'extraction de motifs séquentiels sont proposés dans la littérature, nous pouvons citer SPADE (Zaki 2001), GPS (Srikant & Agrawal 1996) et PrefixSpan (Pei et al., 2001). Comme Apriori, la majorité des algorithmes d'extraction de motifs séquentiels se basent sur la fouille des itemsets fréquents vu dans la section 2.2.1.1. Chacun des algorithmes cités précédemment, représente une approche d'extraction particulière. SPADE par exemple, fait de la génération de candidats à partir d'un format vertical de la base de transactions où chaque itemset contient les identifiants de la séquence et de l'événement des occurrences des itemsets présents dans la base *(<itemset : (identifiant_séquence, identifiant_événement)>*). GPS suit presque la même approche mais l'extraction se fait à partir d'un format horizontal (*<Identifiant_séquence : séquence_itemsets>*) (comme le format de Table 2-1). Enfin, PrefixSpan adopte plutôt une approche de comptage de chemins d'une structure de nœuds d'items (pattern growth method) qui se rapproche de la méthode FP-growth pour la découverte de règles d'association (Cf. section 2.2.1.1.1).

Masseglia a fait un état de l'art intéressant sur les méthodes d'extraction de motifs séquentiel (Masseglia et al., 2004).

#### 2.2.1.2. Le classement et prédiction

Selon S. Tufféry (Tufféry 2007), « le classement estime la valeur d'une variable *à expliquer* par d'autres variables du même individu appelées *cibles* ou *explicatives*. Si la variable à expliquer est qualitative alors la technique est appelée Classement et si elle est continue, elle est appelée Prédiction ». Le classement consiste à analyser de nouvelles données et à les affecter à des classes prédéfinies ou modélisées au préalable. Le classement et prédiction sont des problèmes supervisés d'analyse de données qui sont souvent utilisés pour prédire des valeurs ou des libellés de classes.

Plusieurs techniques, approches et méthodes de classement et de prédiction existent, comme les arbres de décision, les réseaux bayésiens, le raisonnement à base de règles, les algorithmes génétiques, la régression linéaire/non linéaire et le support vecteur machine. Elles utilisent toutes deux étapes : une étape d'apprentissage et une étape de classement. La première étape se focalise sur la construction du classifieur qui va caractériser chaque classe importante par rapport aux données d'apprentissage (variables cibles) et l'attribut des libellés de classes (variable à expliquer). La deuxième étape consiste quant à elle à classifier les nouvelles données selon le classifieur défini dans la première étape.

Bien qu'elles utilisent les mêmes phases, les méthodes peuvent donner des résultats différents qui peuvent être liés aux caractéristiques des données analysées. La comparaison entre les différentes méthodes est parfois donc nécessaire. La comparaison peut être sur la capacité qu'a la méthode à bien classer les nouvelles données (précision), la vitesse de génération et d'utilisation du classifieur, la capacité à donner de bons résultats malgré la présence de données bruitées (aberrantes, manquantes, etc.) et la scalabilité[47].

Les arbres de décision sont parmi les méthodes les plus utilisées à cause des règles explicites fournis par le classement, sa faible indépendance avec l'échantillon de données (robustesse), sa facilité de compréhension et d'interprétation par les utilisateurs. Cette méthode présente les résultats du classement sous forme d'une structure en organigramme. Chaque nœud de l'arbre est un attribut et chaque branche reliant les nœuds est une valeur de l'attribut ou un intervalle de valeurs si l'attribut est continu

[47] Capacité d'un système de monter en charge.

(Figure 2-3). L'intuitivité de cette méthode a fait son succès, plusieurs algorithmes implémentant cette méthode ont été développés. Certains de ces algorithmes ne produisent que des arbres binaires dont chaque nœud de l'arbre produit deux branches. Parmi les algorithmes les plus connus et utilisés, on trouve CART (Breiman et al., 1984), ID3[48] (Quinlan 1979; Quinlan 1982; Quinlan 1986) et C4.5 qui est le successeur de ID3. Contrairement à ID3, C4.5 prend en compte les attributs continus et les valeurs manquantes.

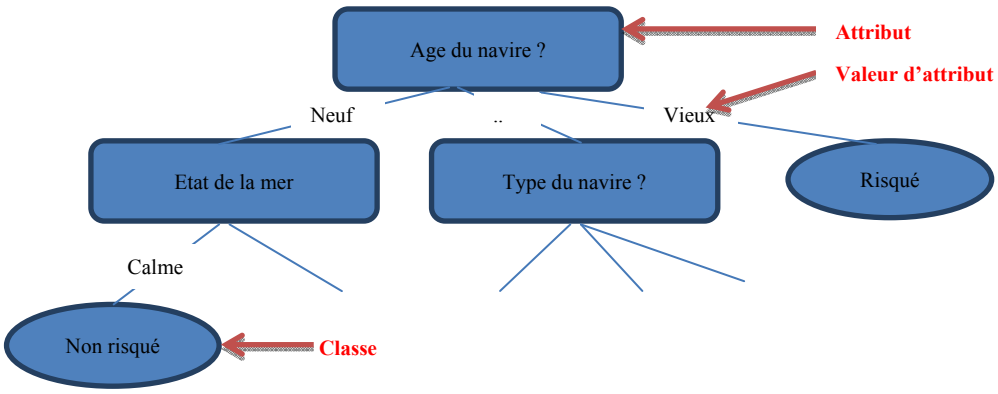



Figure 2-3 : Exemple d'arbre de décision

La majorité des algorithmes de classement par arbre de décision prennent en entrée, les données d'apprentissage avec la distinction de l'attribut de libellé de classe et la méthode de sélection d'attributs les plus discriminant. Cette méthode heuristique de sélection permet de déterminer les critères de séparation entre les classes en utilisant des mesures de sélection comme l'index de Gini et le gain d'information. L'objectif est de trouver les attributs qui séparent le mieux les classes pour avoir un modèle en arbre le plus concis possible.

Prenons l'exemple de l'algorithme ID3 (Algorithme 2-2), qui suit le principe suivant : il prend en entrée un échantillon de données E, un ensemble d'attributs A, les libellés de classes c et donne en sortie une racine d'un arbre de décision. L'algorithme calcule récursivement sur les attributs A non encore choisis par l'algorithme, l'entropie de Shannon pour trouver quel attribut maximise le gain d'information. Quand cet attribut est identifié, un nœud est créé. Si ce nœud est terminal il est étiqueté à Classe, sinon un autre attribut est sélectionné, un sous arbre est créé et l'algorithme passe à un nœud

[48] Interactive Dichotomiser 3

suivant qui n'a pas encore été exploré. Quand l'un des critères d'arrêt est vérifié, il s'arrête et retourne le nœud racine qui est le premier nœud de l'arbre.

L'entropie de Shannon d'un attribut S ayant i valeurs d'attribut $S(x_1,...,x_i)$, sachant que le nombre de classes est égal à $j$, $C(c_1,...,c_j)$ est calculée comme suit :

$E(S) = -\sum_i P(x_i) \sum_j P(c_j/x_i) \log (P(c_j/x_i))$, $P(c_j/x_i)$ est la fréquence relative de la classe j dans le segment S.

**Algorithme ID3**

**Entrée** : E (échantillon de données), A (ensemble d'attributs),
C (classes)
**Sortie** : Racine (Racine de l'arbre de décision)

**Début**
initialiser l'arbre à vide;
**Si** tous les exemples de E ont la même classe c
**Alors** étiqueter la racine par c;
**Sinon si** l'ensemble des attributs A est vide
**Alors** étiqueter la racine par la classe majoritaire dans E;
**Sinon** soit a le meilleur attribut choisi dans A;// *celui qui maximise le gain*
étiqueter la racine par a;
**Pour** toute valeur v de a
Construire une branche étiquetée par v;
Soit Eav l'ensemble des exemples tels que e(a) = v;
Ajouter l'arbre construit par ID3(A-{a}, Eav, c);
**Finpour**
**Finsinon**
**Finsinon**

**Retourner** racine;
**Fin**

Algorithme 2-2 : Algorithme ID3 proposé par Quainlan (Quinlan 1986).

### 2.2.1.3. Le groupement

Le groupement est utilisé depuis toujours dans le subconscient humain pour distinguer les différents éléments qui composent le monde qui l'entoure (Han & Kamber 2006)[49]. Un schéma conceptuel est créé et amélioré continuellement par apprentissage, pour distinguer par exemple un navire de pêche et un navire de plaisance et un risque d'un non-risque. Appelé Clustering en anglais, l'objectif de ce problème est de savoir grouper, dans les mêmes classes, les enregistrements (individus) qui semblent similaires. Contrairement au classement, le groupement n'a pas de variables à expliquer, c'est un

[49] Page 384

problème non-supervisé de fouille de données. En effet, le groupement n'a pas besoin de connaître à l'avance les classes auxquelles appartiennent les individus mais il peut les définir. Le classement par contre  a besoin d'une phase de groupement si les classes des individus ne sont pas connues à l'avance.

Plusieurs techniques peuvent être utilisées pour la découverte de clusters. Ces techniques sont de plusieurs catégories : les méthodes de partitionnement, les méthodes hiérarchiques, basées sur la densité, basées sur les grilles, sur les modèles, etc. De nombreux algorithmes efficaces ont été proposés dans la littérature optimisant les performances et la qualité des classes obtenues dans de grandes bases de données (Tufféry 2007). Voici les plus couramment utilisés, organisés par méthodes :

- méthodes par partitionnement (K-means, K-medoids, CLARANS, EM, etc.),
- méthodes hiérarchiques (DIANA, BIRCH, CURE, etc.),
- méthodes de densité (DBSCAN, OPTICS, etc.),
- méthodes de grilles (STING, WaveClusters, Clique),
- méthodes par modèle (Réseaux de neurones, etc.).

Les méthodes par densité sont particulièrement intéressantes car elles permettent de gérer le bruit (données aberrantes, etc.), de découvrir des clusters ayant des formes arbitraires et d'identifier automatiquement le nombre de clusters.

Pour bien comprendre le concept de clustering par densité, nous présentons un exemple de cette méthode (Figure 2-4). Si le nombre de voisins d'un objet dépasse un seuil minimum (dans l'exemple MinPts >=3) alors l'objet est appelé noyau (centre du cluster). o, r, p, s, m, q sont considérés comme des noyaux. Les voisins d'un objet sont à des distances inférieures au rayon qui est une distance de voisinage définie par l'utilisateur. q est considéré comme directement accessible par densité à partir de m et indirectement accessible à partir de p mais p n'est pas indirectement accessible à partir de q car ce n'est pas un noyau (nombre de voisins < MinPts). Cette accessibilité est appelée Density-reachable. o, r, s forment une connexion ou une liaison dite de densité (Density-connected) forme le cluster.

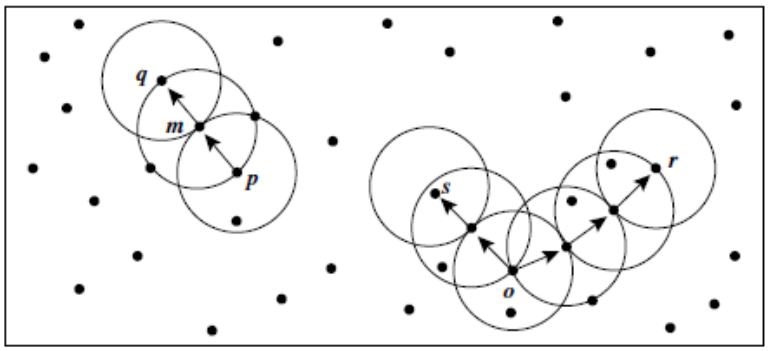



Figure 2-4 : Exemple de construction de clusters à base de densité
(Miller & Han 2009)[50].

Après avoir présenté les concepts de base du clustering de densité à partir d'un exemple, nous détaillerons l'un des algorithmes les plus utilisé, à savoir DBSCAN. Cet algorithme a été présenté pour la première fois lors de la conférence KDD en 1996 par Ester (Ester et al., 1996). L'algorithme commence par sélectionner arbitrairement un point p parmi l'ensemble des points, trouve tous les points qui lui sont accessibles à partir d'une distance radius (Eps). Si p a un nombre de voisins dépassant MinPts alors il est considéré comme noyau et un cluster ayant p comme centre est créé. Si p est un point frontière et il y a pas de point qui lui sont accessibles alors le point suivant est visité. Le processus continue jusqu'à ce que tous les points soient traités. Pour plus de détails sur DBSCAN, l'algorithme est présenté dans l'Algorithme 2-3.

**Algorithme DBSCAN**

```
Entrée : D (les données), eps (radius ou distance de voisinage), MinPts
(minimum d'objets voisins pour constituer une classe)
Sortie : C (les classes)
Début

C = 0
pour chaque point P non visité des données D
marquer P comme visité
PtsVoisins = epsilonVoisinage(P, eps)
si tailleDe(PtsVoisins) < MinPts
mark P as NOISE
sinon
C++
expandCluster(D, P, PtsVoisins, C, eps, MinPts)
```

[50] Page 419

```
Fin
expandCluster(D, P, PtsVoisins, C, eps, MinPts)
ajouter P au cluster C
pour chaque point P' de PtsVoisins
si P' n'a pas été visité
marquer P' comme visité
PtsVoisins' = epsilonVoisinage(D, P', eps)
si tailleDe(PtsVoisins') >= MinPts
PtsVoisins = PtsVoisins U PtsVoisins'
si P' n'est membre d'aucun cluster
ajouter P' au cluster C

epsilonVoisinage(D, P, eps)
retourner tous les points de D qui sont à une distance inférieure à
epsilon de P
```

Algorithme 2-3 : Algorithme DBSCAN.

#### 2.2.1.4. Les Séries chronologiques

Les Séries chronologiques permettent d'identifier les séquences similaires à une portion de données, de prévoir et de déterminer les causalités à partir des bases de données de séries temporelles. Ces bases de données peuvent êtres des bases de données séquentielles (voir section 2.2.1.1.2) ou de valeurs (mesures) obtenues pour des intervalles de temps, comme le cas des mesures de température.

Les bases de données séquentielles ne sont pas forcément des bases de données de séries temporelles car elles peuvent contenir des événements séquentiels non étiquetés dans le temps.

L'objectif des Séries temporelles est de chercher dans de grandes quantités de données, des motifs similaires, réguliers, cycliques, les comportements (tendance), les impulsions, etc. Cela permet de modéliser le mouvement en le décomposant en une série de mouvements basiques (tendance, saisonnalité, cyclicité, irrégularités, horizontalité, etc.) et faire des prédictions de mouvements futures. La tendance correspond à une croissance ou une décroissance de la variable avec le temps, la saisonnalité est caractérisée par une série qui change selon un facteur saisonnier (mois, jour de la semaine, saison), la cyclicité est analogue à une loi saisonnière mais la longueur du cycle est supérieure à un an et ne se répète pas nécessairement à des intervalles de temps réguliers, l'irrégularité est un comportement qui présente une variabilité par rapport au comportement global et enfin l'horizontalité correspond à une stationnarité de la série où les données ne représentent aucune tendance. C'est dans un objectif de prise de décision

et de prédiction des comportements d'un système au cours du temps que les séries temporelles sont décomposées, analysées, expliquées et modélisées.

Plusieurs applications utilisent les séries temporelles comme le marketing (voir Figure 2-5 ), la médecine, la chimie et les finances.

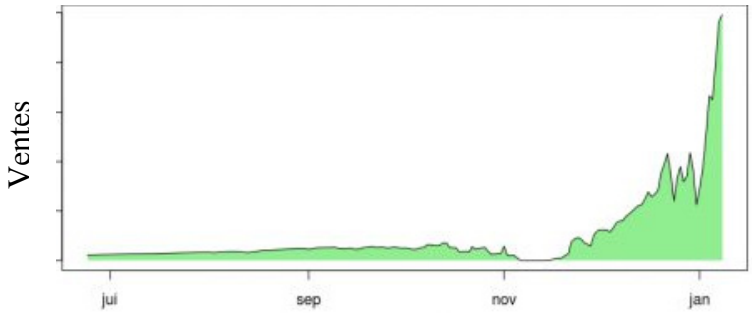



Figure 2-5 : Exemple d'une série chronologique représentant des bénéfices de ventes.

### 2.2.1.5. Analyse d'aberrations

Les aberrations, appelées aussi outliers (terme anglais) sont des données qui ne suivent pas le comportement ou le modèle général. Les outliers doivent être manipulés avec prudence car ils peuvent décrire une erreur ou une variabilité importante dans le comportement du système étudié. Ces outliers sont soit supprimés pour qu'ils n'influencent pas les tendances globales, ou au contraire mis en évidence s'ils véhiculent une information utile à un domaine d'application particulier. Il y a énormément de domaines qui s'intéressent à ce problème de détection d'aberrations.

L'œil humain est très efficace pour la détection d'outliers à partir de visualisations s'il n'y a pas énormément de données affichées. Mais dès que la charge visuelle augmente et les contraintes de temps se manifestent, l'œil atteint ses limites (Han & Kamber 2006). Ainsi il y a un réel besoin d'automatiser cette détection. Dans cet objectif, différentes méthodes informatiques ont été proposées dans la littérature. Ces méthodes utilisent différentes approches comme les statistiques, les mesures de distances, la déviation et la densité (Han & Kamber 2006)[51]. Chacune de ces approches va être détaillée dans les sous-sections ci-après.

[51] Page 452

### *2.2.1.5.1. Détection d'outliers par les statistiques*

Plusieurs méthodes statistiques pour la détection des valeurs aberrantes existent que ce soit sur des données univariées ou multivariées. Pour les méthodes univariées, nous pouvons citer la plus naïve qui consiste à définir un seuil à partir duquel les valeurs sont considérées comme outliers, les méthodes algébriques qui calculent les distances qui séparent les valeurs de la distribution de son centre (moyenne, médiane, quantiles, etc.), les méthodes graphiques ou visuelles comme les Box Plot (Cf. section 1.5.1) (Vasyechko et al., 2005).

Il peut s'avérer parfois utile d'analyser plusieurs variables (multivarié) pour détecter les outliers. Dans certaines variables, la détermination univariée de valeurs aberrantes peut aboutir à une mauvaise classification (Vasyechko et al., 2005).

### *2.2.1.5.2. Détection d'outliers par mesure*

Les outliers ici sont des données qui n'ont pas assez de voisins. Les voisins sont des données se trouvant à un certain seuil de distance comme le radius de DBSCAN (Cf. section 2.2.1.3). La distance peut être métrique comme la distance euclidienne ou non-métrique comme le coût de déplacement (prix, temps, consommation de carburant, etc.). Plusieurs algorithmes de détection d'outliers basés sur la distance existent comme Index-based algorithm, Nested-loop et Cell-based algorithm.

La détection des outliers par les distances et les statistiques posent quelques difficultés quant à l'analyse de données non uniformément distribuées et/ou ayant des densités variables (Figure 2-6). L'exemple donné pas J. Han (Han & Kamber 2006)[52], illustre bien la différence entre les résultats des différents types de méthodes. Il représente la distribution de valeurs par croisement de deux variables non spatiales.

[52] Page 456

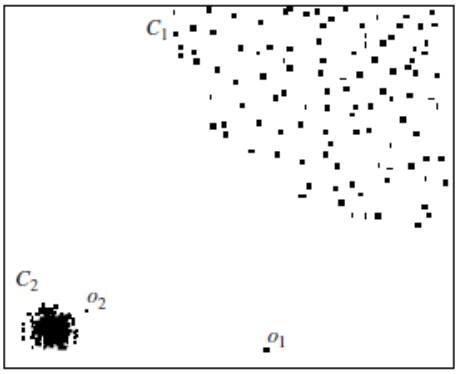



Figure 2-6 : Détection des outliers. Il est évident que o1 et o2 sont des outliers mais o2 ne sera pas considéré comme outliers dans la méthode basée sur les statistiques et les distances car o2 est dans le voisinage des objets de C2. La distance qui sépare o2 de c2 est petite
(Han & Kamber 2006).

### *2.2.1.5.3. Détection d'outliers par déviation*

La détection des outliers basée sur la déviation (Zhang & Feng 2009) se focalise sur l'identification des caractéristiques principales des groupes dans l'objectif de découvrir les objets outliers qui sont déviés de ces caractéristiques des groupes (Han & Kamber 2006). Cette méthode n'utilise ni les distances, ni les tests statistiques.

### *2.2.1.5.4. Détection d'outliers basée sur les densités*

Les outliers sont considérés dans ce paragraphe comme des petits groupes d'objet ou de données non denses et isolés. L'approche Local Outlier Factor (LOF) (Breunig et al., 2000) est souvent utilisée pour ce type de détection.

Le clustering est utilisé aussi pour ce type de détection en identifiant les objets isolés qui ne font partie d'aucun cluster (Han & Kamber 2006)[53].

## 2.2.2. La fouille de données spatiales

Selon ESRI[54], 80% des données possèdent une composante spatiale[55]. La prise en compte de cette composante en explorant les données et les relations spatiales par la

[53] Chapitre 7, page 383
[54] Editeur de système de gestion de l'information géographique
[55] http://www.esri.com/library/whitepapers/pdfs/reveal-more-value.pdf

fouille de données, ouvre des perspectives intéressantes pour la découverte de nouveaux modèles. En effet, la simple visualisation des données sur une cartographie peut permettre la découverte de connaissances comme l'avait démontré le Dr. John Snow (Berche 2007)[56] en trouvant des corrélations entre la localisation des cas de choléra qui avait touché une région du nord de Londres en 1854 et la localisation des puits d'eau. Le Dr. J. Snow est considéré comme l'initiateur de l'analyse cartographique. Cette découverte visuelle des connaissances peut être utilisée sur un volume de données limité mais elle doit être automatisée pour de grandes quantités de données. Cette automatisation est supportée par la fouille de données spatiales.

Le domaine de la fouille de données spatiale est un domaine à part entière qui a été amorcé par les premiers travaux de Koperski, J. Han (Han et al., 1997) et Ester (Ester et al., 1997). Ce domaine s'intéresse à la découverte de modèles dans une base de données spatiales (Committee 2003). La principale caractéristique de ce domaine est sa prise en compte de la dimension spatiale et des relations entre les objets (Chelghoum & Zeitouni 2004). Les objets étudiés sont des thèmes rassemblant les objets de même type. Ces thèmes ne sont rien d'autre que des tables avec un attribut de localisation où les interactions entre les objets sont représentées par des prédicats et des tables de distances.

Les techniques de fouille de données classiques ne peuvent pas supporter directement la composante spatiale dans l'exploration automatique des données pour trois raisons :

- Les données géospatiales sont plus complexes et contiennent des objets, des nombres et des catégories (hétérogénéité). Les objets ont différents types de données, plusieurs types de géométries possibles (point, lignes, polygone) et les relations entre ces objets sont souvent implicites (relations topologiques, relations de distance, direction). Les relations topologiques peuvent être entre toutes les combinaisons possibles de géométries (points*points, point*ligne, etc.).

- Les techniques classiques utilisent des données explicites mais dans le géospatial, les données peuvent être construites à la volée (par exemple l'Overlaping) et les relations spatiales sont la plupart du temps implicites.

[56] Page 70

- Les données de fouille de données classique sont traitées indépendamment des autres (on suppose qu'elles sont indépendantes) alors que dans la fouille de données spatiales, les données sont spatialement dépendantes les unes des autres.

Les méthodes de fouille de données classiques doivent alors être adaptées pour supporter la composante spatiale. Des travaux s'intéressent à l'adaptation de ces méthodes en transformant par exemple des problèmes de la fouille de données spatiales en des problèmes de fouille de données multi-tables (Chelghoum et al., 2006). Contrairement à une représentation *individu-enregistrement* qui est souvent utilisée dans la fouille de données classique, la fouille de données multi-tables se base sur une représentation des données sous forme d'individus dans une table associée potentiellement à plusieurs enregistrements dans des tables secondaires (Dhafer et al., 2012). Dans la littérature, parmi les méthodes classiques principalement adaptées, nous trouvons, les règles d'association, les statistiques et la classification. Les méthodes de fouille de données spatiales sont la plupart du temps scindées en deux types (Aufaure et al., 2000) :

- **Les méthodes exploratoires ou mono thématiques** :

Ces méthodes s'appliquent à un seul thème géographique et permettent d'identifier les écarts et/ou les similarités entre les objets. Ces méthodes sont souvent basées sur les analyses statistiques et elles sont scindées à leur tour en trois familles :

1. Description synthétique : autocorrélation, généralisation, densité et lissage.
2. Spécificités locales : autocorrélation locale, analyse factorielle locale
3. Groupement de données : Clusters de points, clusters de trajectoires, etc.

- **Les méthodes décisionnelles ou multithématiques :**

Ces méthodes s'appliquent à plusieurs thèmes géographiques dans le but d'expliquer les écarts et les caractéristiques des groupements. Donc elles peuvent être vues comme une deuxième phase après l'exploration. Parmi ces méthodes nous pouvons citer la caractérisation, les règles d'association et la classification spatiale adaptées à la fouille de données spatiales. Ces méthodes sont souvent basées sur de l'induction et les techniques de bases de données spatiales.

La section 2.2.1.3 explique l'intérêt des méthodes de groupement par densité par rapport aux autres. Le groupement de données spatiales, et plus particulièrement, le groupement de positions permet d'organiser les positions en classes d'une façon à ce que les objets similaires soient dans les mêmes classes et les objets dissimilaires soient dans des classes différentes. Les méthodes de groupement basées sur la densité séparent les espaces denses (les clusters) des espaces moins denses (le bruit) (Figure 2-7). Le cluster de densité a été défini par J. Han comme étant un ensemble maximal de points connectés par la densité (Han & Kamber 2006)[57].

Figure 2-7 : Exemple de clusters de densité où chaque cluster a une grande densité de points (G. Gasso & P.Leray, 2013)[58].

## 2.2.3. La fouille de données d'objets mobiles

De plus en plus d'objets mobiles sont équipés de capteurs générant un nombre important de positions qui sont reçues, stockées et visualisées dans le but de surveiller l'évolution des systèmes (Etienne et al., 2010). L'application de la fouille de données aux historiques de positions d'objets mobiles ouvre de nouvelles perspectives intéressantes. En effet, cela va permettre d'identifier les comportements normaux et anormaux d'objets en mouvement et faire des prédictions ou des classements. D'un autre point de vue, le déplacement pose des problèmes aux modèles de données géospatiales et à la fouille de données spatiales. En effet, il est difficile d'identifier et de catégoriser les objets mobiles sur des trajectoires (Committee 2003).

Plusieurs travaux de recherche se sont intéressés à la fouille de données d'objets en mouvement. Parmi ces travaux nous pouvons citer les travaux de l'équipe de J. Han (Lee

[57] Page 418
[58] Gilles Gasso et Phillippe Leray, Clustering, INSA Rouen, 2013

et al., 2008a; Li et al., 2010c), de K. Zeitouni (Kharrat et al., 2009) et Giannotti et Nanni (Giannotti et al., 2007). Dans la littérature, des travaux font la distinction entre la fouille de patterns d'objets mobiles et la fouille de données de trajectoires. On considère dans la suite de ce travail que la fouille de données d'objets mobiles regroupe les méthodes qui prennent en compte la dimension temporelle et celles qui ne s'intéressent qu'à la dimension spatiale.

La distinction qui nous captive est celle qui considère deux types de méthodes de fouille de données : le premier type voit les objets mobiles comme des objets qui se déplacent dans un espace ouvert et le second, comme des objets se déplaçant dans un espace soumis à des contraintes de réseaux.

#### 2.2.3.1. Espace d'évolution ouvert

J. Han propose des méthodes de fouille de données d'objets mobiles pour révéler des mouvements collectifs, des clusters de trajectoires (Lee et al., 2007), détecter des trajectoires aberrantes (Lee et al., 2008a), détecter des périodicités dans les déplacements (Li et al., 2010b) et faire des classification d'objets selon leur trajectoire (Lee et al., 2009). Avant Li, Cao (Cao et al., 2007) s'est penché sur la question des périodicités et a proposé une méthode pour les découvrir. Parmi les méthodes de clustering de trajectoires, on trouve la détection des ensembles d'objets se déplaçant en convois comme la méthode proposé par l'équipe de Jeung (Jeung et al., 2008a).

##### *2.2.3.1.1. Détection de trajectoires aberrantes*

Les trajectoires aberrantes représentent des mouvements qui ne ressemblent pas au comportement général. Les trajectoires aberrantes sont de deux types : les trajectoires qui ne suivent pas les mêmes chemins que les autres objets, et les sous-trajectoires qui ne suivent pas la même tendance que les autres sous-trajectoires se trouvant dans son voisinage (voir la sous-trajectoire aberrante TR3 de la Figure 2-8). Les sous-trajectoires peuvent contenir une ou plusieurs partitions, la partition étant l'unité la plus petite d'une trajectoire après la position.

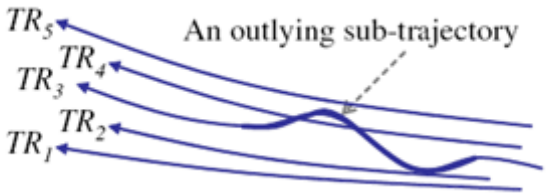



Figure 2-8 : Exemple d'une sous-trajectoire aberrante (Lee et al., 2008a).

La plupart des algorithmes proposés dans la littérature pour la comparaison de trajectoires considèrent des trajectoires complètes sans les partitionner en sous trajectoires. Cette approche empêche la détection de partitions de trajectoires outliers (Figure 2-8). Lee (Lee et al., 2008a) présente un algorithme de détection de trajectoires et de sous-trajectoires aberrantes appelé TRAOD (TRAjectory Outlier Detection). L'algorithme travaille en deux phases, il commence par une phase de partitionnement des trajectoires en sous trajectoires puis il enchaine par une deuxième phase de détection des outliers basée sur une mesure de distance et de densité (Figure 2-9). L'objectif est de pouvoir détecter les trajectoires et les sous trajectoires outliers dans les partitions denses et non denses.


Figure 2-9 : Illustration du fonctionnement de l'algorithme TRAOD.

***Comment sont effectués les partitionnements des trajectoires ?***

L'algorithme prend en entrée un ensemble de trajectoires sous forme de séquences de positions successives. Le partitionnement de ces trajectoires est basé sur deux approches différentes : une approche basée sur le principe MDL (*Minimum Description Length*), et une approche à deux niveaux.

Le principe MDL (Grünwald et al., 2005) est souvent utilisé dans la théorie de l'information. Le problème de partitionnement est ainsi transformé en un problème MDL. L'intérêt de ce partitionnement est son aptitude à optimiser les données sans avoir besoin de paramètres en entrée comme le cas de la réduction de données spatio-temporelles (Cao et al., 2006). Le partitionnement basé sur MDL cherche à trouver le meilleur compromis entre la longueur de la description de l'hypothèse L(H) et la longueur de la description des données lorsqu'elles sont codées à l'aide de l'hypothèse L(D|H). Le compromis est une fonction qui minimise la somme des deux éléments L(H) et L(D|H). H désigne l'hypothèse et D les données.

Dans l'algorithme TRAOD, l'auteur essaye de minimiser la longueur de la nouvelle trajectoire simplifiée en minimisant au maximum le nombre de positions de la trajectoire (concision) et en respectant le plus possible la représentation de la trajectoire originale (précision). La nouvelle trajectoire simplifiée, doit donc assurer une meilleure représentation de la trajectoire originale à travers ses partitions, et une meilleure concision à travers le nombre de partitions constituant cette nouvelle trajectoire.

Plus il y a de partitions, meilleure sont la représentation et la qualité de détection, mais les performances sont réduites. La complexité algorithmique de TRAOD est de $O(n^2)$, telle que $n$ est le nombre total de partitions des trajectoires.

Dans l'objectif de réduire l'espace de recherche et avoir de meilleures performances, une approche de partitionnement en deux niveaux est proposée : des partitions grossières en premier niveau, puis ; si nécessaire, les partitions grossières sont partitionnées à leur tour, en partitions fines en deuxième niveau (Figure 2-10).

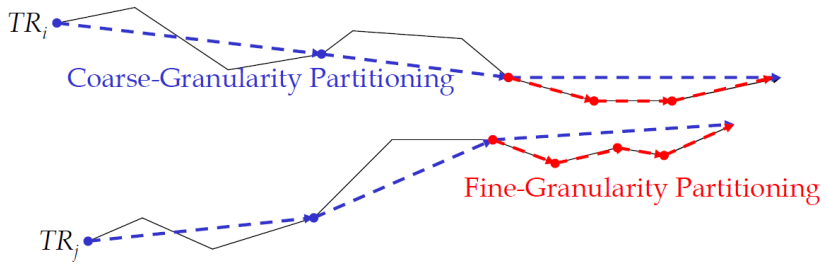
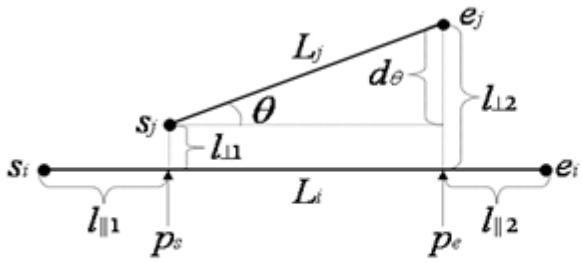



Figure 2-10 : Partitionnement de trajectoires en deux niveaux, t-partions grossières et t-partition fines (Lee et al., 2008a).

L'algorithme TRAOD inspecte les partitions grossières pour partitionner en fines partitions celles qui sont susceptibles d'être outliers. L'identification des partitions pouvant être des outliers est effectuée en se basant sur le concept de borne supérieure (*ub)* et de borne inferieure (*lb)* de la distance entre deux partitions grossières Li et Lj.

Le partitionnement en fines partions n'est pas nécessaire si la borne inférieure est supérieure à la distance entre Li et Lj ($lb > D$). Le partitionnement en fines partitions est par contre effectif si la borne supérieure est inférieure ou égale à la distance entre Li et Lj ($ub \leq D$).

La distance entre deux partions est calculée sur la base d'une distance perpendiculaire $d_{\perp}$, parallèle $d_{\parallel}$ et angulaire $d_{\theta}$ (Figure2-11) qui sont pondérées par rapport à des poids $w(w_{\perp}, w_{\parallel}, w_{\theta})$ pour donner plus de poids, score ou d'importance à une distance par rapport à une autre. Par exemple, dans l'une des expérimentations menées par Lee et son équipe (Lee et al., 2008a), le poids de la distance angulaire est fixé à cinq fois la distance parallèle et la distance perpendiculaire pour la découverte de trajectoires aberrantes dans le déplacement d'ouragans.




$$d_{\perp} = \frac{l_{\perp 1}^2 + l_{\perp 2}^2}{l_{\perp 1} + l_{\perp 2}}$$
$$d_{\parallel} = \mathrm{MIN}(l_{\parallel 1}, l_{\parallel 2})$$
$$d_{\theta} = \|L_j\| \times \sin(\theta)$$

$$dist(L_i, L_j) = w_{\perp} d_{\perp} + w_{\parallel} d_{\parallel} + w_{\theta} d_{\theta}$$

Figure 2-11 : Calcul de distance entre deux partitions de trajectoires intégrant la distance perpendiculaire, parallèle et angulaire (Lee et al., 2008a).

Le *lb* et *ub* sont calculés comme suit :

$$lb(L_i, L_j, dist) = w_{\perp} \cdot lb(Li, Lj, d_{\perp}) + w_{\parallel} \cdot lb(Li, Lj, d_{\parallel}) + w_{\theta} \cdot lb(Li, Lj, d_{\theta})$$
$$ub(L_i, L_j, dist) = w_{\perp} \cdot ub(Li, Lj, d_{\perp}) + w_{\parallel} \cdot ub(Li, Lj, d_{\parallel}) + w_{\theta} \cdot ub(Li, Lj, d_{\theta})$$

Le calcul des bornes supérieures (*ub*) et inférieures (*lb*) des distances perpendiculaires, parallèles et angulaires sont présentées ci-après :

- **Distance perpendiculaire** :

$$lb(L_i, L_j, d_\perp) = MIN\ (l_{\perp 1}, l_{\perp 2}) - (\max l_\perp(L_i) + \max l_\perp(L_j))$$
$$ub(L_i, L_j, d_\perp) = MAX\ (l_{\perp 1}, l_{\perp 2}) - (\max l_\perp(L_i) + \max l_\perp(L_j))$$

Telle que, $l_{\perp 1}$ et $l_{\perp 2}$ sont des distances euclidiennes des deux extrémités de la partition $L_j$ sur leur projection sur $L_i$,

$\max l_\perp(L_i)$ : Distance perpendiculaire maximale entre une partition grossière et ses partitions fines,

$\max l_\theta(L_i)$ : Angle maximal entre une partition grossière et ses partitions fines.

La figure suivante (Figure 2-12) représente les différentes composantes de la formule définie ci-dessus et montre comment ils sont calculées.

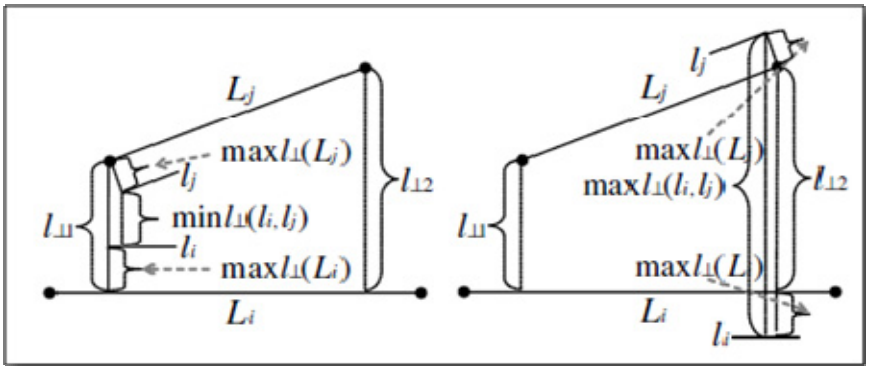



Figure 2-12 : Illustration des distances qui rentrent dans le calcul des composantes *lb* et *ub* de la distance perpendiculaire (Lee et al., 2008a).

- **Distance parallèle :**

$lb(L_i \text{ et } L_j, d_{//}) = 0$ *si les partitions $L_i$ et $L_j$ sont superposées ou jointes*
$= d_{//}(L_i, L_j)$ *si elles sont disjointes.*

$ub(L_i \text{ et } L_j, d_{//}) = MAX(len(L_i), len(L_i))$ *si les partitions $L_i$ et $L_j$ sont jointes*
$= len(L_i)+len(L_j)-d_{//}(L_i, L_j)$ *si elles sont superposées*
$= len(L_i)+len(L_j)+d_{//}(L_i, L_j)$ *si elles sont disjointes*

- **Distance angulaire :**

$$lb(L_i, L_j, d_\theta) = MIN(\min len(L_i), \min len(L_j)) * \sin(\theta - \max\theta(L_i) - \max\theta(L_j));$$
$$ub(L_i, L_j, d_\theta) = MIN(\max len(L_i), \max len(L_j)) * \sin(\theta + \max\theta(L_i) + \max\theta(L_j));$$

Telle que θ est l'angle entre 2 partitions grossières $L_i$ et $L_j$ et $max\theta(L_i)$ est l'angle maximal entre la partition grossière $L_i$ et ses partitions fines.

### *Comment s'effectue la détection ?*

L'algorithme prend en entrée trois paramètres qui sont la distance entre les partitions de trajectoires voisines (D), la proportion de trajectoires voisines pour ne pas être une partition aberrante (p) et la proportion de la longueur des outliers dans une trajectoire pour qu'elle soit aberrante (F). La détection des outliers est donc basée sur un seuil de distance (D) et un seuil de densité (p). L'algorithme compare la distance entre les partitions des trajectoires et la distance fournie par l'utilisateur (D). L'algorithme TRAOD est présenté ci-après (Algorithme 2-4).

- Une partition est dite outlier si elle n'a pas assez de partitions voisines similaires (voir Figure 2-13)

*Nombre trajectoires voisines* $\leq$ *(1-p)* nombre total de trajectoires,*

- Une trajectoire est dite outlier si elle contient une proportion de longueur suffisante de partitions outliers :

$$\frac{\text{la somme des longueurs des partitions outliers de la trajectoire i}}{\text{la somme des longueurs de toutes les partitions de la trajectoire i}} \geq F$$

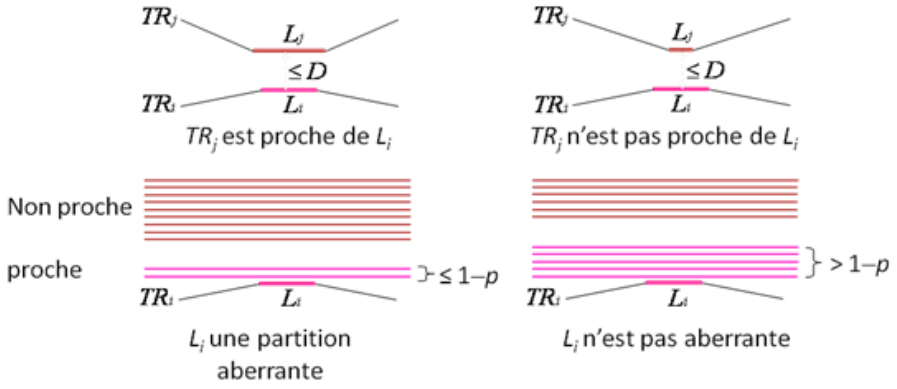



Figure 2-13 : Détection des outliers basée sur la distance et la densité (source Lee[59]).

[59] http://dm.kaist.ac.kr/jaegil/slides/icde08.ppt

Cette approche ne prend pas en compte l'hétérogénéité de la distribution des trajectoires : les partitions se trouvant dans des régions denses ont relativement un nombre de partitions voisines plus grand que celles se trouvant dans des régions éparses et ont donc moins de chances d'être détectées comme étant outliers. Un facteur d'ajustement $adj(L_i)$ est multiplié par le nombre de trajectoires voisines pour ajuster la densité locale par rapport à la densité moyenne. Ce facteur est un rapport entre la moyenne de toutes les densités des partitions et la densité de la partition $L_i$.

**Algorithme TRAOD (TRAjectory Outlier Detection)**

Entrée : I = {TR1, …, TRn} (ensemble de trajectoires)
Sortie : O = {O1, …, Om} (ensemble de trajectoires outlier avec leurs partitions associées)
**Début**
/* Partitioning Phase */
Pour chaque TR ∈ I
Partitionner TR en un ensemble de segments de lignes L
Mettre L dans un ensemble D
/* Detection Phase */
Pour chaque P ∈ D
Vérifier si P est une partition outlier
Pour chaque TR ∈ I
Retourner les TR ayant suffisamment d'outlier
**Fin**

Algorithme 2-4 : Algorithme TRAOD.

### *2.2.3.1.2. Clustering de trajectoires*

Le clustering de trajectoires s'inspire du clustering de positions (Cf. section 2.2.2) où les positions sont remplacées par des trajectoires. L'objectif de ce clustering est de regrouper les trajectoires individuelles en des trajectoires représentatives du mouvement général. Deux types d'algorithmes existent : ceux qui traitent les trajectoires dans leur ensemble comme ceux proposés par Gaffney (Gaffney et al., 2006), et l'algorithme TRACLUS de L. Lee (Lee et al., 2007) qui propose de partitionner les trajectoires en segments de lignes avant de regrouper les partitions similaires. L'algorithme TRACLUS (TRAjectory CLUStering) permet de faire du clustering en deux phases, une phase de partitionnement et une phase de groupement (voir Figure 2-14). Il commence par une phase de partitionnement des trajectoires en petites partitions, basée sur un algorithme

glouton[60] appelé *Approximate Trajectory Partitioning*. Cet algorithme est proposé par le même auteur de TRACLUS pour le partitionnement de trajectoires car le partitionnement par MDL coûte cher en performance. Une approche considérant que les optimaux locaux peuvent être des optimaux globaux va alors réduire le temps de partitionnement. L'algorithme glouton de partitionnement proposé reste basé sur le problème MDL vu dans la section précédente. Les partitions résultantes de la simplification des trajectoires par *Approximate Trajectory Partitioning* sont utilisées dans la deuxième phase pour grouper celles qui sont similaires. Le groupement des partitions de trajectoires est basé sur une extension de l'algorithme DBSCAN.

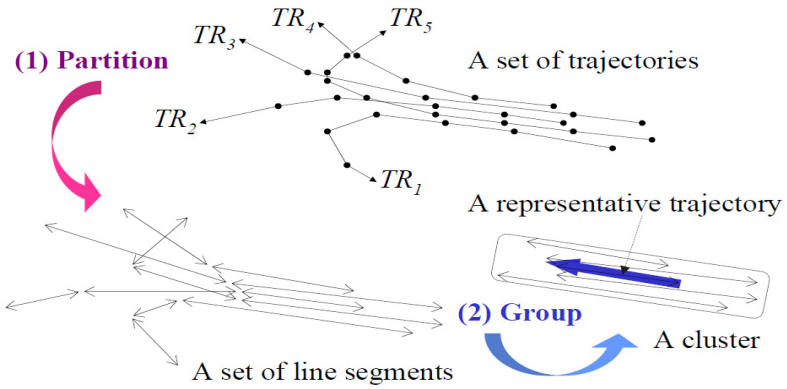



Figure 2-14 : Les deux phases de l'algorithme TRACLUS : Partition et groupement des partions pour la découvertes des clusters de trajectoires (Lee et al., 2007).

Comme DBSCAN, TRACLUS utilise un paramètre de distance de voisinage (ε-voisinage) tel que les partitions se trouvant à une distance inférieure de ε-voisinage sont des partitions voisines ($N\varepsilon(Li) = \{Lj \in D \mid dist(Li, Lj) \leq \varepsilon\}$). Si le nombre de partitions voisines d'une partition Li dépasse le nombre minimal de partitions ($|N\varepsilon(Li)| \geq MinLns$) alors c'est une partition noyau. Une trajectoire représentative est donc créée pour chaque cluster de trajectoires (Trajectoire rouge sur la Figure 2-15) à partir de ces partitions noyaux qui sont les segments principaux des ensembles de partitions voisines. Des points sont créés pour chaque début et fin d'une partition et le cluster est créé en suivant une

[60] Appelé greedy algorithm en anglais, c'est un algorithme approximatif qui essaye d'approcher un optimum global en choisissant des optimaux locaux à chaque itération, Thierry Mautor, PRISM, UVSQ, 2009.

ligne de balayage. La direction est calculée comme une moyenne des vecteurs de direction des partitions voisines.

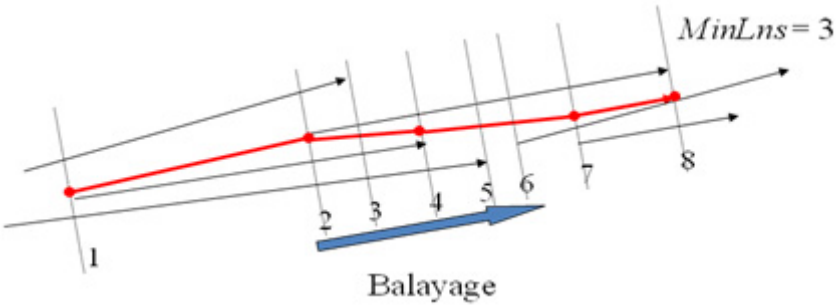



Figure 2-15 : Exemple de construction d'un cluster de trajectoires pour MinLns =3.

Les définitions de DBSCAN ont été adaptées pour TRACLUS. Une partition Li est dite directement joignable par densité à partir de Lj par rapport à ε-voisinage et MinLns, si Li ∈ Nε(Lj) et |Nε(Lj)| ≥ MinLns et les partitions sont joignables par densité avec la propriété de fermeture transitive de la jointure directe de densité. Li est connecté à Lj par la densité, s'il existe Lk tel que Li et Lj sont joignables par densité de Lk. Les ensembles connectés par densité forment les clusters de trajectoires. L'algorithme de TRACLUS est présenté ci-après (Algorithme 2-5).

**Algorithme TRACLUS**

Entrée : I (ensemble de trajectoires) {TR1,…,TRnum_traj}
Sortie : O (ensemble de clusters) {C1,…, Cnum_clus},
R (ensemble représentative des trajectoires)
**Début**

/*Phase de partitionnement*/
**Pour chaque** TRi ∈ I
Partitionnement des trajectoires avec une méthode approximative
Obtenir un ensemble L de segments à partir des résultats du partitionnement
Mettre L dans D

/*Phase de groupement*/
Clustering des segments de trajectoires de D
Obtenir un ensemble O de clusters à partir des résultats de clustering
**Pour chaque** C ∈ O
Construire les trajectoires représentatives
Obtenir un ensemble R des trajectoires représentatives à partir des résultats
**Fin**

Algorithme 2-5 : Algorithme TRACLUS.

Il est à remarquer que la trajectoire représentative du clustering est sensible à l'hétérogénéité des partitions de trajectoires : directions différentes et décalage non perpendiculaire entre les débuts et les fins des partitions. La Figure 2-16 illustre bien l'effet de ce décalage sur la forme de la trajectoire représentative.

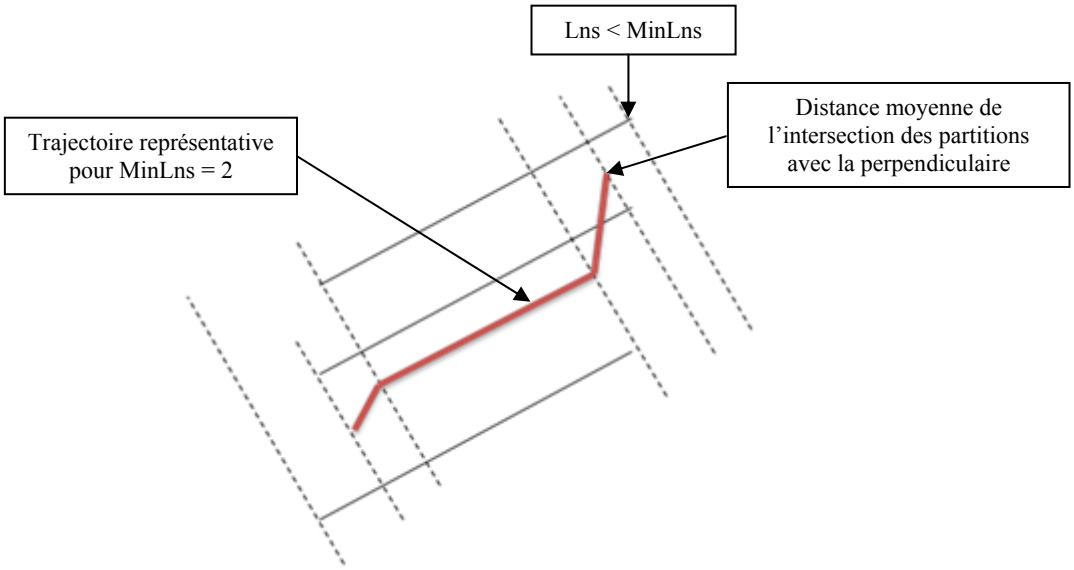



Figure 2-16 : Illustration du calcul de la trajectoire représentative à partir de trois partitions parallèles avec des départs décalés.

### *2.2.3.1.3. Classement de trajectoires*

Comme vu dans la section 2.2.1.2, le classement permet de modéliser les classes caractéristiques des groupes de données pour permettre le classement de nouvelles données. Le classement de trajectoires est une extension de ce problème aux données mobiles. Parmi les algorithmes proposés dans la littérature pour le classement des trajectoires, nous pouvons citer *Trajectory Classification Using Hierarchical Region-Based and Trajectory-Based Clustering* (Lee et al., 2008b). TRACLASS pose le postulat que les objets similaires fréquentent les mêmes régions et ont les mêmes comportements.

Cet algorithme s'exécute en trois phases, il commence par une phase de partitionnement de trajectoires, puis de groupement et ensuite de classement. Après le partitionnement de trajectoires, il trouve les régions rectangulaires majoritaires où le nombre des partitions de la même classe dans une région est supérieur ou égal à $\psi$ (Figure 2-17- (1) (2)). Une fois les régions denses identifiées (groupement basé sur les régions), les trajectoires se trouvant à l'extérieur des régions sont partitionnées et regroupées (groupement basé sur les trajectoires) en prenant en compte les régions

trouvées au préalable (Figure 2-17- (3) (4)). Deux groupes sont dits connectables s'ils partagent suffisamment de trajectoires (fraction des trajectoires ≥ χ). Le groupement est donc un groupement en deux niveaux, basé sur les régions et basé sur les trajectoires.

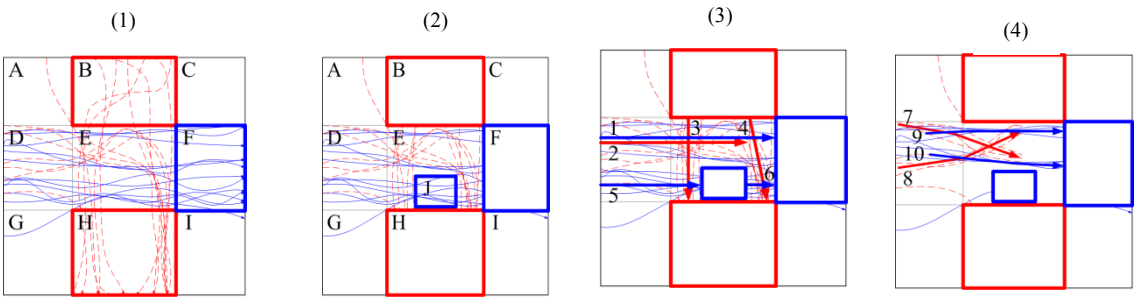



Figure 2-17 : Exemple de groupement de régions (1) (2) puis de trajectoires (3) (4) (Lee et al., 2008b).

Un cluster basé trajectoires est un ensemble de partitions qui sont connectées par densité et qui appartiennent à la même classe comme illustré sur la Figure 2-18.

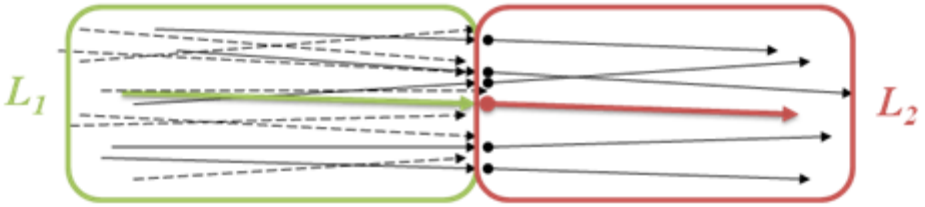



Figure 2-18 : Le groupement de partitions de trajectoires selon le libellé de chaque classe (Lee et al., 2008b).

Comme il est représenté dans le processus de classification de TRACLASS (Figure 2-19), après la phase de groupement basé sur les régions et sur les trajectoires, le groupement basé sur les deux est effectué. La phase de groupement régions s'arrête soit quand il n'y a plus de régions hétérogènes, soit quand la taille minimale d'une région est atteinte. En ce qui concerne le groupement basé trajectoires, il s'arrête quand tous les groupes sont détectés. Le classifieur mis en œuvre à la fin du processus de classification de TRACASS peut être utilisé pour classer de nouveaux comportements.

L'idée de la phase de classification consiste à convertir chaque trajectoire en un vecteur caractéristique dont chaque entrée est soit un cluster basé sur les régions, sur les trajectoires soit une caractéristique numérique. Après avoir défini ces vecteurs caractéristiques, le classificateur sera construit par une méthode de généralisation des

classificateurs linéaires appelée *Support Vector Machines* (SVM). Le SVM « *Support Vector Machines* » est un ensemble de techniques d'apprentissage supervisé dans des problèmes de discrimination et de régression.

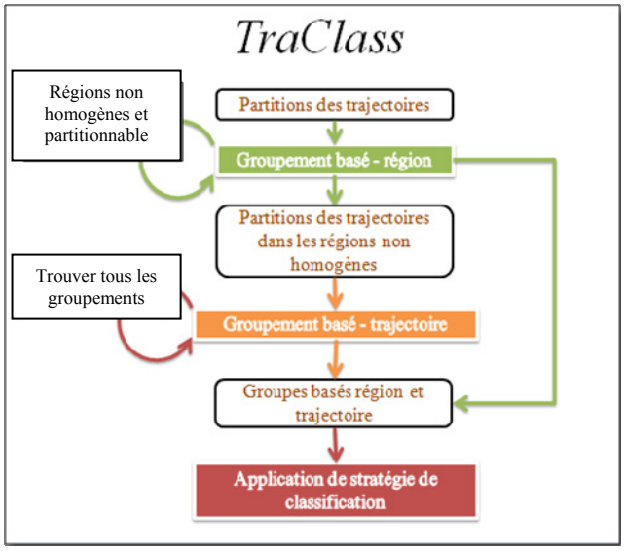



Figure 2-19 : Processus de classement de l'algorithme TRACLASS (Lee et al., 2008b).

Prenons un exemple d'application de cet algorithme sur un ensemble de trajectoires de navires évoluant entre un port A et un port B (voir Figure 2-20) qui a permis la construction d'un classifieur de trajectoires.

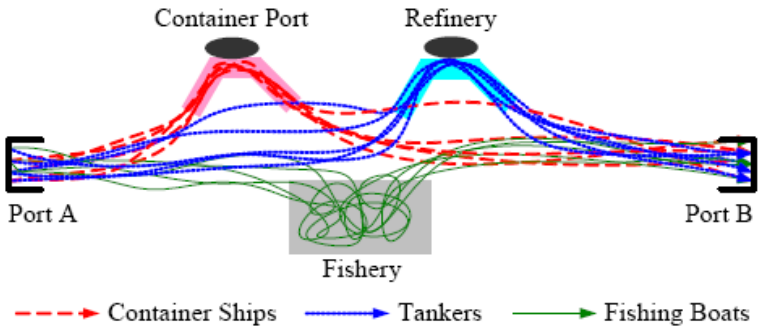



Figure 2-20 : Exemple de classement de trajectoires de navires (Lee et al., 2008b).

L'algorithme détecte d'abord les régions ayant le plus de trajectoires homogènes (régions autour de Container Port, Rafinery et Fishery), puis découvre les groupes de trajectoires partagées entre les régions (mouvements communs des classes de trajectoires). Les régions et mouvements communs de chaque classe de trajectoire sont utilisés comme caractéristiques de trajectoires pour discriminer les types des navires

selon leur trajectoire. Le classifieur construit à partir des vecteurs caractéristiques des trajectoires de navires va permettre d'associer à chaque nouvelle trajectoire l'une des classes suivantes : Container Ships, Tankers et Fishing Boats.

#### *2.2.3.1.4. Détection de Clusters d'objets mobiles*

Les clusters d'objets mobiles ou de mouvements permettent de détecter des objets qui se déplacent conjointement pendant un intervalle de temps long (Kalnis et al., 2005). Ce sont des clusters spatiaux successifs dans le temps ayant un seuil d'objets en communs suffisamment grand pendant une durée de temps exacte ou minimale selon l'approche choisie. Il existe dans la littérature plusieurs études de clusters d'objets mobiles comme les clusters en mouvement (Kalnis et al., 2005), les Flocks (Li et al., 2011) (Gudmundsson & Van Kreveld 2006) (Benkert et al., 2008), les Convois (Jeung et al., 2008a; Jeung et al., 2008b) et les Swarm (Li et al., 2010a). Les Flocks sont souvent utilisés dans plusieurs applications pour la détection automatique de sous-ensembles d'objets mobiles suivant des trajectoires voisines pendant une durée de temps prédéfinie. L'algorithme Flock se base sur la détection de clusters de rayons fixes ayant un nombre d'objets mobiles minimal pendant des instants de temps successifs t=1, 2,3. Une extension intéressante de l'algorithme Flock est l'algorithme Convoy. En effet, il suppose que les objets se déplaçant ensemble peuvent sortir du groupe ou y entrer à tout moment (Figure 2-21). Par comparaison à l'algorithme Flock, l'algorithme Convoy permet la détection de clusters de formes arbitraires en intégrant la notion de liaison de densité (Cf. section 2.2.1.3). L'algorithme Convoy demande de transformer les trajectoires discrètes en trajectoires continues dans le temps. Les trajectoires traitées par L'algorithme Convoy doivent êtres continues pour pouvoir faire des groupements à n'importe quel temps.

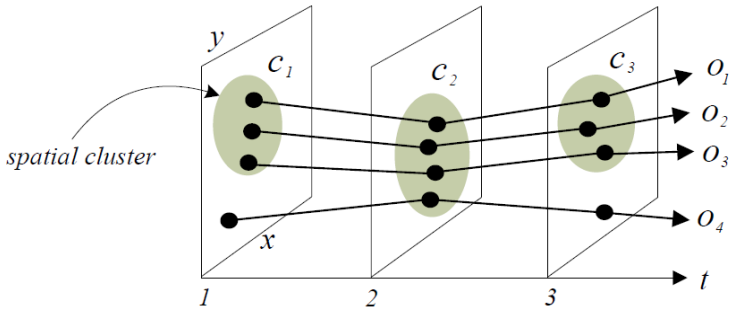



Figure 2-21 : Détection de convois pour un intervalle de temps et un nombre minimal d'objets égal à 3.

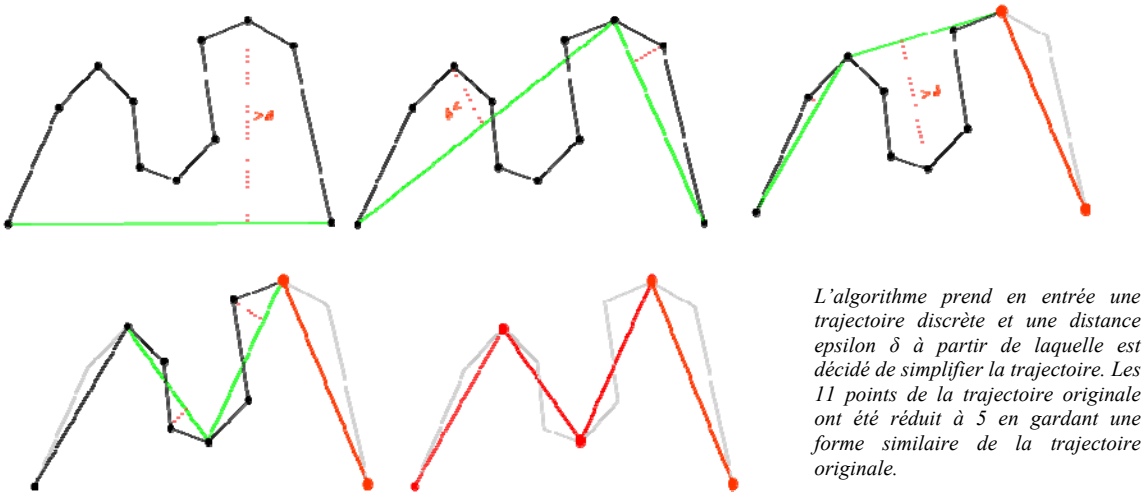
La détection de convois, peut avoir plusieurs applications pratiques comme la détection des troupeaux d'animaux, la détection des manifestations dans les foules et la découverte de navires navigant ensemble.

En s'inspirant des études de détection de clusters d'objets mobiles, plusieurs algorithmes ont été proposés comme les algorithmes de base MC1, MC2 et MC3 développé par Kalnis (Kalnis et al., 2005), BFE (Basic Flock Evaluation) (Vieira et al., 2009) pour la détection de Flock et les algorithme CMC (Coherent Moving Clusters), CuTS (Convoy Discovery using Trajectory Simplification), CuTS+ et CuTS* (Jeung et al., 2008a, Jeung et al., 2008b) pour la découverte de Convoy. L'algorithme CuTS propose une simplification des trajectoires en entrée, les transformant de trajectoires discrètes en trajectoires continues dans le temps. Cette simplification permet aux positions d'êtres définies à tout moment et donc la possibilité de détecter les clusters de mouvement à chaque instant. De plus, cet algorithme offre de meilleures performances que CMC en termes de rapidité de traitement à cause de la simplification des données trajectoires.

Ces algorithmes de détection de convois proposés par l'équipe de H. Jeung différent selon la phase de simplification choisie. CuTS utilise l'algorithme de Douglas Peucker (Douglas and Peucker, 1973) pour la compression de données et la simplification des trajectoires par généralisation. Il permet de ne garder que les positions caractéristiques ou significatives. Formellement, étant donné une trajectoire représentée par la polyligne suivante $o = <p_1, p_2, ..., p_T>$, et un paramètre de tolérance δ donné en entrée, l'objectif de la simplification de trajectoires est de proposer une autre polyligne $o'$ contenant moins de points que la trajectoire originale et s'écartant d'au plus δ de la trajectoire originale o.

Comme illustré sur la Figure 2-22, l'algorithme de Douglas Peucker (DP) relie avec un segment de droite les deux points extrêmes de la trajectoire. Il calcule la distance entre le point le plus éloigné de la trajectoire et ce segment de droite. Si cette distance est supérieure à epsilon δ (seuil de précision), il considère le point le plus éloigné comme le nouveau point extrême de la trajectoire, sinon c'est le segment de droite qui devient la trajectoire simplifiée. La trajectoire est coupée récursivement jusqu'à ce que la condition de distance ne soit plus vérifiée pour toute la trajectoire.

Le fonctionnement de l'algorithme de Douglas-Peucker est illustré à la Figure 2-22.



*L'algorithme prend en entrée une trajectoire discrète et une distance epsilon δ à partir de laquelle est décidé de simplifier la trajectoire. Les 11 points de la trajectoire originale ont été réduit à 5 en gardant une forme similaire de la trajectoire originale.*

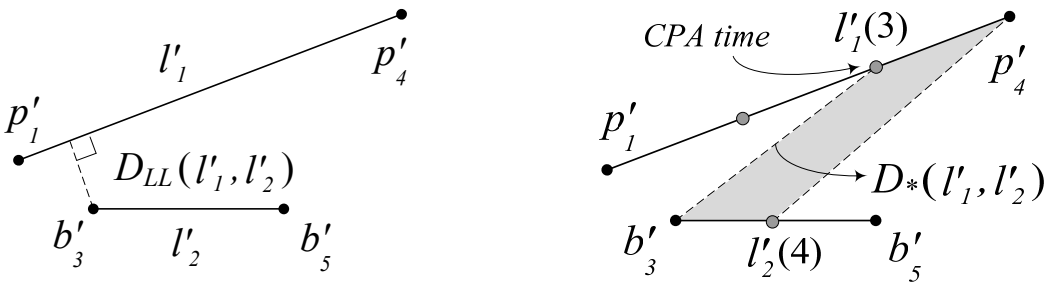
Figure 2-22 : illustration de l'algorithme Douglas-Peucker sur une simplification de trajectoire (source Jeung[61]).

Pour une simplification plus rapide, l'algorithme CuTS+ intègre l'algorithme DP+ qui identifie les points extrêmes dichotomiquement pour arriver plus rapidement à la simplification optimale. Cependant l'algorithme DP original et son extension de simplification ne prennent pas en compte la dimension temporelle dans la simplification, ce qui a incité H. Jeung et son équipe à proposer CuTS*. CuTS* intègre DP* (Meratnia

[61] Présentation de Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S. Jensen, Heng Tao Shen, "Discovery of Convoys in Trajectory Databases", PVLDB, 1(1):1068-1080, 2008

& De By 2004) qui calcule les distances entres les positions de trajectoires et les segments de droite en considérant le temps (Figure 2-23).

Cet algorithme offre une meilleure efficacité selon les études et tests effectués par H. Jeung. La Figure 2-23 montre la différence entre le calcul de distance entre l'algorithme de DP classique et son amélioration DP* intégrant la dimension temps dans la simplification. Le calcul de distance classique utilisé par DP entre deux segments de trajectoires est basé sur la plus petite distance perpendiculaire entre les deux segments originaux et représentatifs. Cette distance DLL (l'1, l'2) est représentée sur la Figure 2-23-(1).




(1) Algorithme DP (2) Algorithme DP*

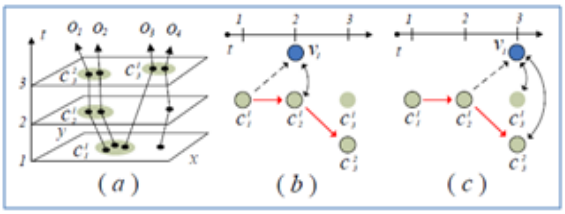
Figure 2-23 : Différence entre le calcul de distance des algorithmes DP et DP*. Dans (1) une distance perpendiculaire est calculée et dans (2) une distance entre 2 positions ayant le même timestamp (Jeung et al., 2008a).

Les algorithmes CMC, CuTS, CuTS+ et CuTS* prennent en entrée, un ensemble de trajectoires de N objets, un seuil de distance e minimum pour le groupement, un nombre minimum d'objets m pour former un convoi et une durée de temps k de déplacement de convoi. L'objectif de ces algorithmes est la découverte rapide et efficace des groupes d'objets se déplaçant ensemble en se basant sur la densité de trajectoires par rapport à la distance e et le nombre de trajectoires m durant k instants consécutifs. Même si ces algorithmes ont le même principe, chacun à ses spécificités.

CMC (Coherent Moving Cluster) est un algorithme basé sur une simple technique de détection de convois. Tout d'abord, l'algorithme utilise une interpolation linéaire des

positions de trajectoires pour compléter celles qui manquent par des positions virtuelles à des instants t donnés. Ensuite, un groupement des objets est effectué à chaque instant t en utilisant la connexion par densité (Cf. 2.2.2 du chapitre 2) pour détecter les clusters possibles. Le rayon utilisé est e. Après avoir détecté ces clusters, l'algorithme cherche les convois candidats qui contiennent des positions communes entre des clusters consécutifs pendant k instants. Un convoi est actuel si au moins les k clusters candidats partagent un nombre suffisant de trajectoires communes. Pour que le nombre soit suffisant, le nombre de trajectoires partagées par les clusters candidats doit être supérieur à m.

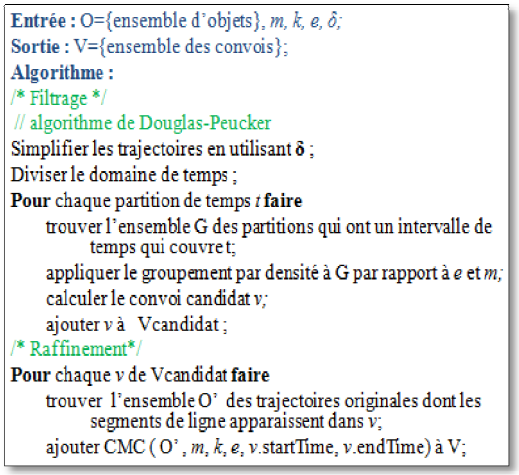
La Figure 2-24 illustre un cas d'exécution de l'algorithme CMC.




| Timestamp | Clusters | Candidate set $V$ |
|---|---|---|
| $t_1$ | $c_1^1$ | $v_1 = c_1^1$ |
| $t_2$ | $c_2^1$ | $v_1 = c_1^1 \cap c_2^1$ |
| $t_3$ | $c_3^1, c_3^2$ | $v_1 = c_1^1 \cap c_2^1 \cap c_3^2, v_2 = c_3^1$ |

Figure 2-24 : Exemple illustrant l'exécution de l'algorithme CMC pour m=2 ; k=3 ; e=◄► .

- CuTS (Convoy discovery Using Trajectory Simplification) est une amélioration de l'algorithme CMC pour réduire sa complexité. CuTS est composé de deux phases, une phase de filtrage et une phase de raffinement (Algorithme 2-6). Dans le filtrage, les trajectoires sont simplifiées en utilisant l'algorithme de Douglas Peucker (DP) qui utilise comme paramètre la tolérance δ définie par l'utilisateur. Les partitions de trajectoires trouvées après simplification sont regroupées pour calculer les convois candidats. Dans la phase de raffinement, l'algorithme CMC est repris avec une amélioration. Avec la simplification des trajectoires, il n'y a pas besoin de calculer les interpolations pour compléter les positions de trajectoires manquantes.



**Entrée :** O={ensemble d'objets}, *m*, *k*, *e*, *δ*;
**Sortie :** V={ensemble des convois};
**Algorithme :**
/* Filtrage */
// algorithme de Douglas-Peucker
Simplifier les trajectoires en utilisant **δ** ;
Diviser le domaine de temps ;
**Pour** chaque partition de temps *t* **faire**
trouver l'ensemble G des partitions qui ont un intervalle de temps qui couvre t;
appliquer le groupement par densité à G par rapport à *e* et *m*;
calculer le convoi candidat *v*;
ajouter *v* à Vcandidat ;
/* Raffinement*/
**Pour** chaque *v* de Vcandidat **faire**
trouver l'ensemble O' des trajectoires originales dont les segments de ligne apparaissent dans *v*;
ajouter CMC ( O' , *m*, *k*, *e*, *v*.startTime, *v*.endTime) à V;

Algorithme 2-6 : Algorithme CuTS.

- CuTS+ améliore la phase de simplification dans CuTS, en utilisant l'Algorithme de Douglas Peucker plus (DP+). DP+ est plus rapide que DP, il diminue l'erreur en cas de groupement car la tolérance actuelle obtenue par DP+ est toujours plus petite que celle obtenue par DP mais un point faible de cet algorithme est qu'il ne conserve pas bien la forme de la trajectoire originale.
- CuTS* est une extension temporelle de CuTS, il introduit la notion du temps lors de la phase de filtrage, en utilisant la version améliorée de Douglas Peucker de base appelé DP* pour la simplification. Cette notion est aussi intégrée lors du groupement en utilisant une fonction de calcul de distance qui respecte la position de l'objet par rapport à l'axe de temps.

### *2.2.3.1.5. Détection des périodiques*

La découverte de mouvements périodiques est intéressante dans l'analyse de comportements des objets mobiles. La périodicité peut être utilisée par exemple pour la prédiction de mouvements habituels, la détection des événements anormaux et pour résumer les longs historiques de mouvements. Parmi les méthodes récentes qui ont été proposées dans la littérature pour la découverte des mouvements périodiques, on trouve Mining Periodic Behaviors for Moving Objects (Li et al., 2010b). Li et son équipe ont proposé cette méthode dans l'objectif de résoudre le problème de détection des comportements périodiques des objets en mouvements. Ce problème est composé de deux sous-problèmes, à savoir la détection des périodes dans les mouvements et la découverte des comportements périodiques.

Les données sur les mouvements observés sont généralement générées à partir de plusieurs comportements périodiques entremêlés, ayant différentes périodes, associés à plusieurs emplacements de référence et ayant plusieurs formats de dates (mois, jours, heures, etc.). Sans oublier le fait que les objets mobiles n'ont pas souvent les mêmes trajectoires ce qui rend difficile la détection des périodicités et leur prédiction. La plupart des méthodes de détection de périodiques appliquent la transformée de Fourier directement sur les séquences de déplacements. Etant donné que les résultats de la transformée de Fourier sont sensibles au bruit[62], la découverte de périodiques peut échouer. Pour résoudre ce problème, la méthode proposée par l'équipe de Li commence par chercher toutes les localisations de références qui sont des zones fréquemment visitées. L'observation des mouvements à partir de ces localisations de référence permet de faciliter la découverte des périodiques.

L'algorithme développé pour supporter cette méthode est appelé Periodica. L'algorithme est en deux étapes : une étape de détection des périodiques à partir des séquences de mouvements données en entrée et une étape de découverte de comportements périodiques (Algorithme 2-7).

**Entrée :** $Loc=loc_1\ loc_2\ \ldots\ locn$;
**Sortie :** un ensemble des comportements périodiques;

**Algorithme :**
**/* Etape 1 : détection des périodes */**
Trouver les localisations de référence $O=\{o_1, o_2, \ldots, o_d\}$;
**Pour** chaque localisation de réf. **Faire**
  Transformer le mvt. En une séquence binaire;
  Trouver les périodes dans chaque localisation de réf. *oi;*

**/* Etape 2 : Découverte de comportements périodiques */**
**Pour** chaque période *T* **Faire**
  Diviser le mvt. $O_T$ = {localisation de réf. de période *T*};
  Construire la séquence de mvt. symbolisée *S* à partir de $O_T$;
  Trouver les comportements périodiques dans *S*;

Algorithme 2-7 : Algorithme Periodica.

Dans la première étape, l'algorithme commence par identifier les localisations de référence en utilisant une méthode de calcul de densité par noyau (Kernel Density Estimation[63]). Ces localisations de référence vont faciliter la découverte des périodes. En effet, en observant le mouvement à partir de ces localisations, les périodes sont plus faciles à percevoir. Ensuite pour chaque emplacement de référence identifié, l'algorithme cherche toutes les périodes possibles en appliquant la transformée de Fourier et l'auto-corrélation circulaire sur la transformation binaire de la séquence de mouvements effectuée au préalable.

[62] Veut dire ici des trajectoires non utiles à la découverte de comportements périodiques.
[63] KDE est une méthode de calcul de densité non-paramétrique.

Dans la deuxième étape de l'algorithme, pour chaque période T trouvée, l'algorithme segmente le mouvement par ces périodes. Ensuite, il considère seulement les localisations de référence et construit une séquence symbolisée de mouvements à partir de ces localisations. C'est à partir de cette séquence symbolisée que Periodica cherche à découvrir les comportements périodiques en utilisant une méthode de groupement (voir 2.2.1.3). Les clusters identifiés à la fin représentent les comportements périodiques.

Pour bien comprendre l'algorithme Periodica, prenons un exemple simple de découverte de périodicités dans un jeu de données de déplacement d'une abeille (Figure 2-25) présenté par Z. Li[64]. L'utilisation de Periodica sur ce jeu de données va permettre en premier lieu d'identifier les localisations de référence, à savoir dans les ruches et en dehors des ruches, puis de transformer les séquences de mouvements en des séquences binaires par rapport aux zones de référence (voir Figure 2-26).

L'utilisation de la Transformée de Fourier sur les séquences binaires de mouvements permet de détecter la présence de périodiques. Pour avoir les valeurs de ces périodiques, l'Autocorrélation Circulaire est utilisée.

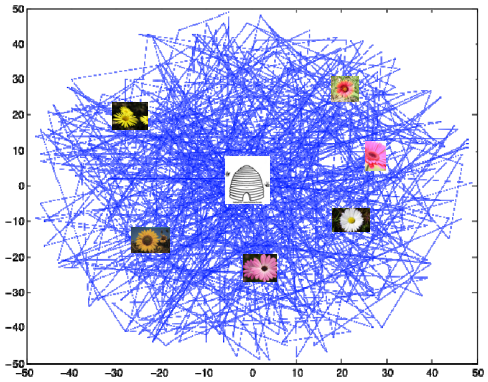



Figure 2-25: mouvement d'une abeille (Li et al., 2010).

[64] http://faculty.ist.psu.edu/jessieli/Publications/KDD10-ZLi-PeriodMovObj.ppsx

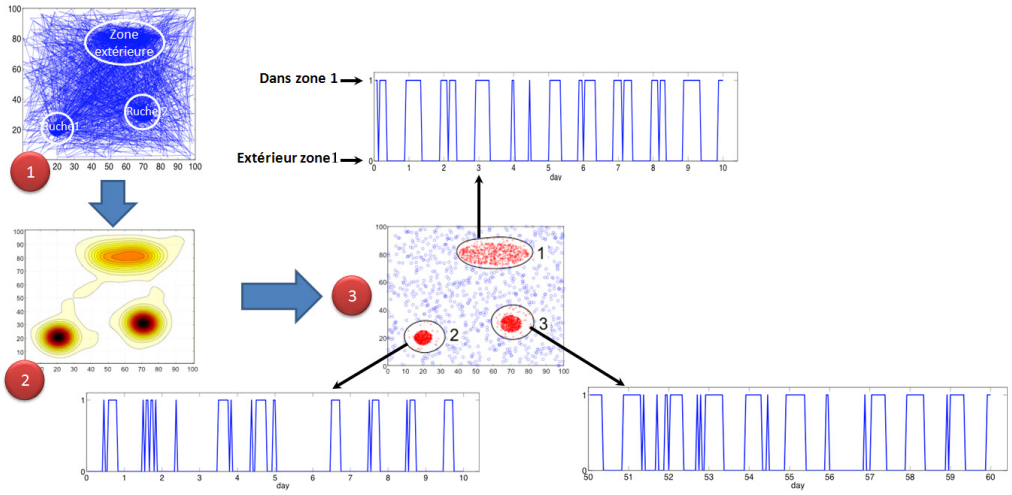



Figure 2-26 : Première étape de Periodica - (1) détection des localisations de références, (2) transformation des mouvements en une séquence binaire, (3) trouver les périodes (Li et al., 2010).

Après avoir identifié la période du comportement de l'abeille qui est de 24h, les mouvements sont symbolisés par les localisations de références, segmentés en périodes de 24h et regroupés par une méthode de classification hiérarchique ascendante (Figure 2-27). Initialement, chaque segment est un comportement et la distance entre les comportements est calculée en utilisant la mesure de dissimilarité divergente de Kullback-Leibler. Le comportement périodique est modélisé sous forme d'une matrice de probabilités de présence dans les zones de références (appelées spot sur la Figure 2-27).

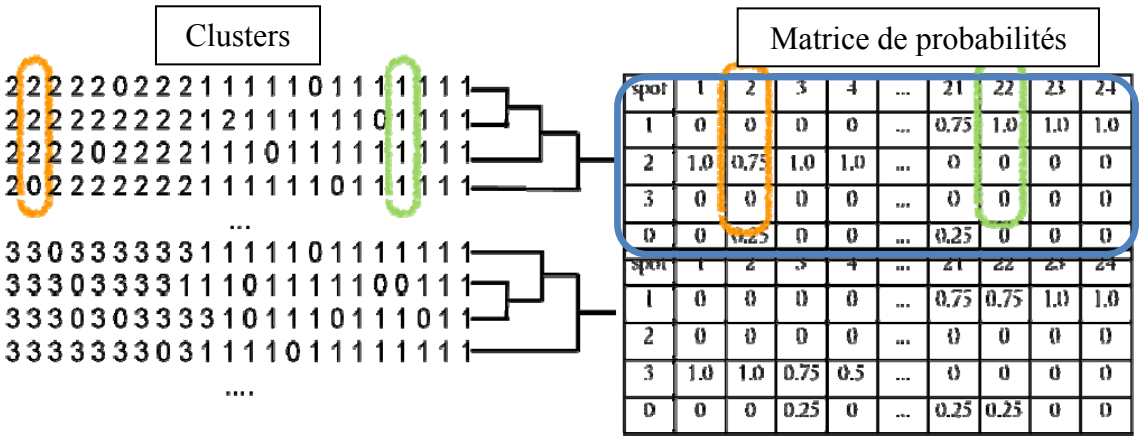



Figure 2-27 : Deuxième étape de Periodica – Clustering des segments de mouvements, construction de la matrice de probabilités et découverte des comportements périodiques (Li et al., 2010).

La découverte du nombre de comportements périodiques est effectuée par l'utilisation de la mesure de pureté du groupement qui est l'erreur de représentation des segments dans un cluster. Deux comportements périodiques ont été identifiés, le premier est un déplacement entre la ruche 1 et la zone extérieure et le deuxième entre la ruche 2 et la zone extérieure. Dans les deux comportements, l'abeille passe 8h dans la ruche et 16h à voler à proximité de la ruche pour chercher de la nourriture.

Un autre exemple d'application intéressante est celui de la découverte d'un comportement migratoire d'un aigle chauve en Amérique du Nord. Les données sur lesquelles l'équipe de J. Han ont fait le test représente le déplacement d'un aigle pendant une durée de 3 ans. La Figure 2-28 montre le résultat de Periodica sur ce jeu de données. La partie (a) de la figure montre les localisations de l'aigle sur Google Earth, la partie (b) illustre les trois localisations de références qui sont détectées par analyse de densité et la partie (c) décrit le comportement périodique de l'aigle. Selon ce résultat, l'aigle chauve reste à la localisation 1 (spot 1) de décembre à mars, il commence à se déplacer vers la localisation 2 (spot 2) en mars, il reste dans cette localisation jusqu'à la fin du mois de mai, après il migre vers la localisation 3 (spot 3) en été. Ensuite, à partir de septembre il revient vers la localisation 2 et y reste de mi-octobre à mi-novembre puis revient à la localisation 1 et ainsi de suite.

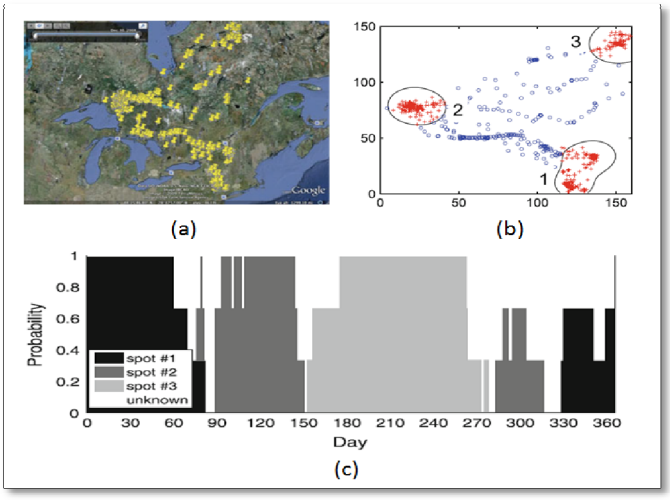



Figure 2-28 : Découverte de comportements périodiques dans le déplacement d'un aigle chauve. (a) Les données affichées sur Google Earth. (b) Découverte des localisations de référence. (c) Le comportements périodiques de l'aigle chauve (Li et al., 2010).

#### 2.2.3.2. Espace d'évolution contraint par un réseau

D'autres méthodes de fouille de données d'objets mobiles, comme le clustering de trajectoires, sous-trajectoires (Kharrat et al., 2008a; Kharrat et al., 2009) (Lai et al., 2007) (Meng, 2007) (Chen et al., 2007) et de découverte de patrons de mobilités (Kharrat et al., 2013) sont utilisées pour les objets mobiles dont le mouvement est soumis à des contraintes de trajectoires (réseau routier, réseau ferré, etc.). Ces objets ne peuvent pas se déplacer librement dans l'espace mais doivent suivre un réseau qui influence la forme et la vitesse de leur mouvement (Kharrat et al., 2013). On parle dans ce cas, de trajectoires contraintes par un réseau (Kharrat et al., 2008b). Pour exploiter ces données de mouvements, plusieurs approches de représentations des données d'objets mobiles sont proposées. Parmi ces approches, nous pouvons citer les deux plus communément utilisées. La première se base sur un découpage de l'espace de déplacement en une grille avec des cases de tailles fixes. La deuxième se base sur une représentation sous forme de graphes représentant le réseau sur lequel se déplacent les objets. Cette dernière représentation est plus intéressante car elle permet d'avoir des résultats précis et complets reflétant la réalité du terrain (Lai et al., 2007). Dans la représentation sous forme de graphe, les arrêtes du graphe sont des segments du réseau et les nœuds sont des interconnections entre les segments. Ces segments peuvent être orientés ou non.

Plusieurs applications ont besoin de ce type de méthodes de fouille de données comme la gestion du trafic routier, la détection des congestions du trafic routier et la découverte de bouchons.

Dans notre problématique, les deux approches de fouille de données peuvent être appliquées. En effet, il est possible de considérer les navires comme des navires soumis à des contraintes de routes, des objets se déplaçant dans un espace ouvert ou comme une combinaison des deux. La combinaison peut être par rapport à des parties de l'espace comme par exemple être soumis à des contraintes de routes dans les approches d'entrée/sortie d'un port et considérer que le déplacement est ouvert en pleine mer. Il est possible aussi d'imaginer une combinaison par rapport au type du navire. Les Tankers ont tendance à suivre des routes bien définies alors que les navires de pêches par exemple ont des déplacements spécifiques qui dépendent de plusieurs critères (type de pêche, zones de pêches, météorologie, etc.).

Dans la suite de ce travail, nous avons fait le postulat que le déplacement des navires se fait plutôt dans un espace ouvert non soumis à des contraintes de routes pour plusieurs raisons :

- Les routes maritimes ne sont pas matérialisées. Dans les méthodes destinées aux objets contraints par un réseau, il faut au préalable de l'analyse, construire le réseau de routes,
- Des types de navires se déplacent d'une manière spécifique sans suivre des routes particulières comme les navires de pêche et de plaisance.

### 2.2.4. La fouille de données du trafic de mobiles

Pour pouvoir distinguer ces deux derniers domaines, à savoir la fouille de données d'objets mobiles et la fouille de données du trafic, il faut observer l'objet sur lequel porte l'analyse. Dans la fouille de données d'objets mobiles, l'analyse porte sur le mobile et ses déplacements. Dans la fouille de données du trafic, l'objectif est d'analyser les flux d'objets mobiles qui passent par des segments de réseaux (routier, ferroviaire, etc.). La différence peut être cernée aussi dans les données de capteurs, étant donné que la fouille de données d'objets mobiles utilise des données individuelles issues de capteurs embarqués alors que la fouille de données du trafic, utilise des données du trafic mesurées en différentes parties du réseau.

On enregistre une abondance de données sur le trafic routier, maritime ou aérien. Google Maps par exemple fournit des informations en temps-réel sur la circulation routière comme on le voit sur la Figure 2-29. Il est même possible de faire des prédictions de circulation par jour et par heure en se basant sur les observations passées. Ces données récoltées peuvent êtres exploitées dans plusieurs applications de fouille de données du trafic comme les systèmes de surveillance du trafic (Lu et al., 2006), la découverte des événements de congestion, la planification des routes, la découverte des routes rapides (Awasthi et al., 2005), populaires, etc. L'objectif de ce domaine de fouille de données est de résumer comment

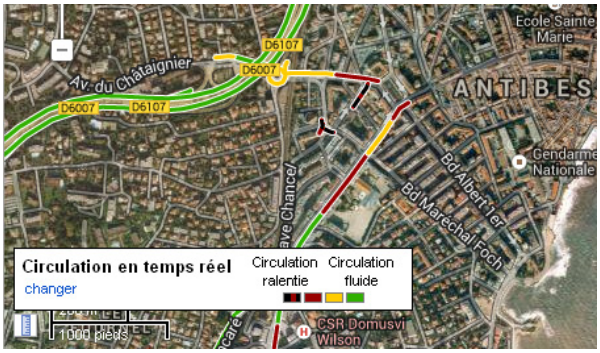



Figure 2-29 : Affichage sur Google Maps du Trafic routier à Antibes en temps-réel.

les objets mobiles se déplacent dans différents intervalles de temps (journée, semaine, etc.), quels sont leurs comportements typiques dans des régions et à des moments spécifiques, etc.

Nous allons exposer dans les sous-sections suivantes, deux méthodes de fouille de données utilisées pour l'exploration de données du trafic : les motifs de trajectoires (T-patterns) et la détection des congestions (T-Flock). Les trajectoires sur lesquelles porte l'exploration sont des séquences de localisations d'objets mobiles et de temps de transition d'un certain point du réseau à un autre.

### 2.2.4.1. Les motifs de trajectoires

Les motifs de trajectoires sont des séquences de zones avec des transitions étiquetées dans le temps : $T = Z_0 \rightarrow_{\alpha 0} Z_1 \rightarrow_{\alpha 1} .. \rightarrow_{\alpha 2} Z_n$, tel que $Z = \{ Z_{0,..,} Z_n\}$ est l'ensemble des zones et $\alpha = \{\alpha_{1,...,} \alpha_n\}$ est l'ensemble des temps de transition entre les zones. Ces motifs de trajectoires sont utilisés pour résumer le comportement du trafic, découvrir les régions de congestions et la prédiction des déplacements futurs, etc. Trajectory Pattern Mining (T-Patterns) proposé par l'équipe *Knowledge Discovery and Data Mining Laboratory (KDD Lab)* de l'Université de Pise (Giannotti et al., 2007), est un algorithme qui permet l'extraction de ce type de motif. Il prend en entrée des données D (latitude, longitude, timestamp), le support de trajectoires, le support de temps et le rayon de voisinage et il retourne un T-Patterns. Le support de trajectoires est le pourcentage de trajectoires qui doivent composer le T-Patterns, le support de temps est le temps minimum d'une transition et le rayon de voisinage $N(X_i,Y_i)$ est le disque englobant les zones denses. Pour comprendre comment l'algorithme fonctionne, un exemple d'extraction d'un motif T-Pattern est représenté sur la Figure 2-30.

L'algorithme suit trois étapes pour extraire les motifs de trajectoires :

- Calculer les RoI (Region of Interest) qui sont les zones denses,
- Convertir les données (x, y, t) en (RoI, t),
- Trouver les séquences de transitions satisfaisant le support de trajectoires et de temps.

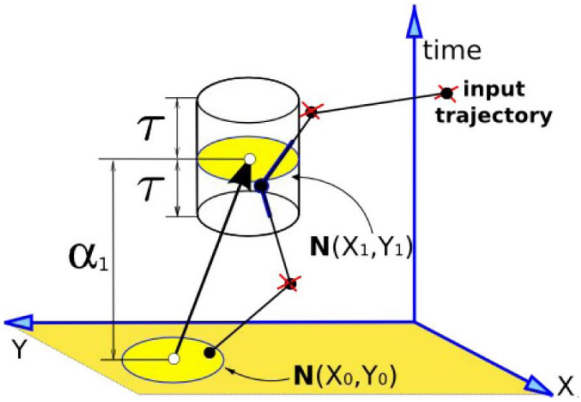



Figure 2-30 : Illustration de la méthode d'extraction d'un motif de trajectoire à partir d'une trajectoire (Giannotti et al., 2007)[65].

### 2.2.4.2. La détection des congestions

La détection de congestions ou des embouteillages à partir des données de trafic peut être vu comme un problème de détection de groupes d'objets se déplaçant ensemble et lentement. L'équipe de Giannotti (Ong et al., 2011) ont proposé une méthode de détection des embouteillages basée sur une extraction des Flocks (Cf. section 2.2.3.1.4). Cette méthode enrichit le concept de Flock par un support de vitesses pour pouvoir identifier les objets qui se déplacent à proximité et ayant des vitesses inférieures au support (T-Flock).

## 2.3. Prototypes d'analyse de comportements

Certaines équipes de recherche qui travaillent activement dans le domaine de la fouille de données, comme l'équipe de Jiawei Han du *Department of Computer Science University of Illinois* et l'équipe de Fosca Giannotti du laboratoire *Knowledge Discovery and Data Mining* de l'université de Pise ont développé des prototypes regroupant les méthodes de fouille de données d'objets mobiles développées en interne. Ces prototypes servent de preuve de concept et de vitrine aux travaux réalisés. Nous décrivons ci-après, les deux prototypes MoveMine et M-Atlas qui ont été développés par ces deux équipes.

[65] Trajectory Pattern Mining, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.187.6467&rep=rep1&type=pdf

## 2.3.1. MoveMine

Le prototype MoveMine (Li et al., 2010c) a été développé dans le cadre d'un projet avec Boeing pour une éventuelle utilisation dans le transport aérien. Un démonstrateur du prototype avait été mise en ligne[66] sans la possibilité de téléchargement du code source ou de son exécutable. Les éditeurs de MoveMine sont tenus à une obligation de confidentialité du projet.

L'architecture système de MoveMine, se compose de trois interfaces (Figure 2-31) : une interface de collection et nettoyage des données, une interface de fouille de données et une interface de visualisation des données sources et des résultats.

Dans l'interface Collection and Cleaning, un ensemble de données de déplacements d'animaux, de véhicules et des événements climatiques sont collectés et prétraités pour permettre leur exploration par les algorithmes de fouille de données. Les données en entrée peuvent contenir des bruits, des imprécisions, des incohérences : c'est pour cela qu'un module de nettoyage est ajouté dans l'architecture du prototype.

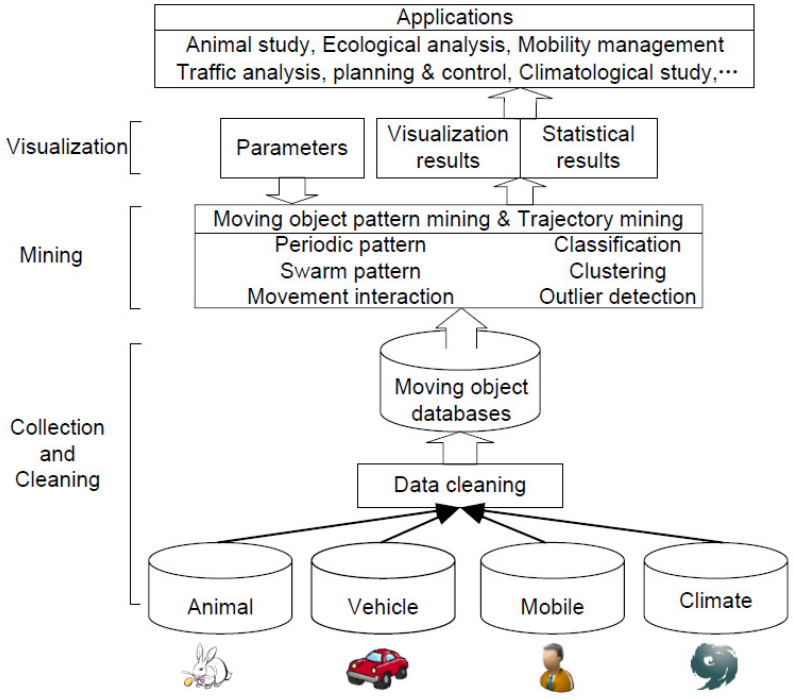



Figure 2-31 : Architecture du prototype MoveMine (Li et al., 2011).

[66] http://dm.cs.uiuc.edu/movemine/

Dans l'interface Mining, un ensemble d'algorithmes de fouille de données d'objets mobiles développés récemment par l'équipe de J. Han ont été intégrés. Un résumé des fonctionnalités de ces algorithmes est représenté sur la Figure 2-32.

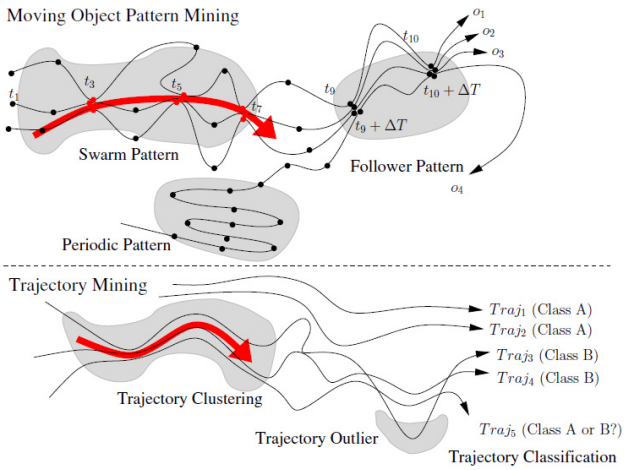



Figure 2-32 : Les fonctionnalités principales de MoveMine

(Li et al., 2011).

A partir de l'interface Visualization, les utilisateurs du système peuvent entrer les paramètres, lancer l'exécution d'un algorithme, afficher les données sur Google Maps, Google Earth et interagir avec les résultats.

Le démonstrateur mis sur internet montre son utilisation pour plusieurs domaines d'application comme l'étude des déplacements d'animaux, l'analyse écologique, la gestion de la mobilité, l'analyse du trafic, l'étude climatique, etc. Il permet de révéler des clusters de trajectoires, de détecter des trajectoires aberrantes (outlier trajectory), d'identifier des comportements périodiques et des mouvements collectifs (Figure 2-33). Le prototype MoveMine permet d'évaluer les algorithmes proposés par l'équipe.

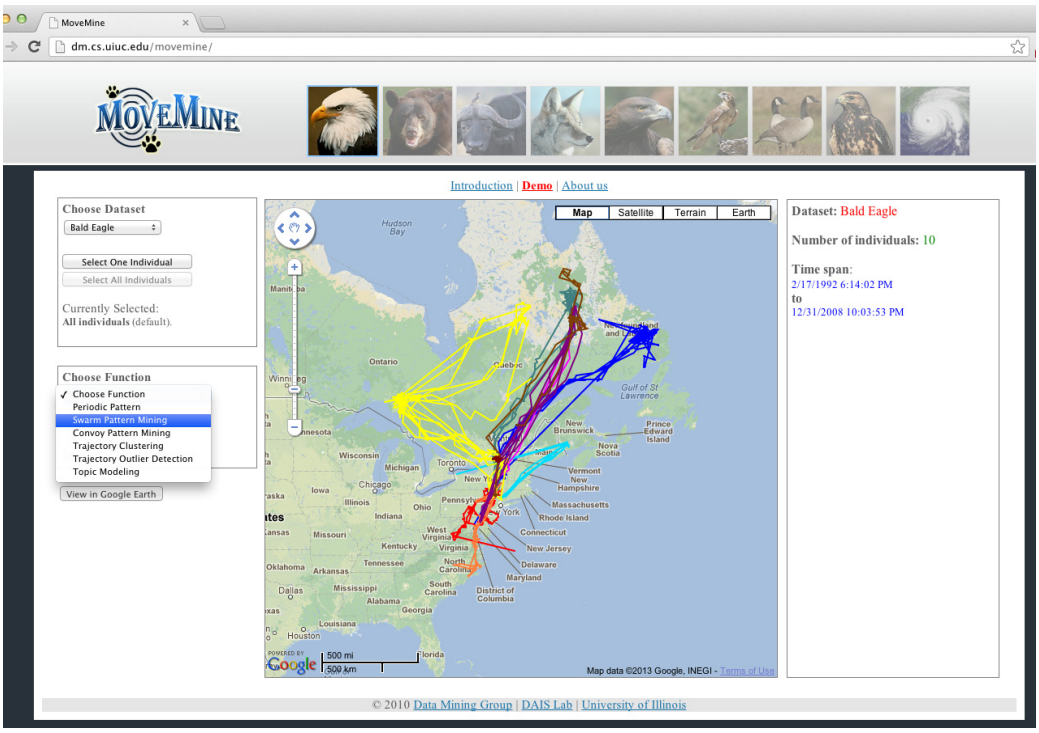



Figure 2-33 : Démonstrateur MoveMine-Exemple de détection de mouvements collectifs à partir d'un jeu de données de déplacement d'aigles (Li et al., 2010c).

## 2.3.2. M-Atlas

M-Atlas est un prototype qui a été développé et mis en téléchargement libre[67] par l'équipe de Giannotti (Giannotti et al., 2011). Il est centré sur le concept de trajectoire sur laquelle porte les différentes analyses. Ce prototype se propose d'analyser le trafic et la mobilité des objets par l'interrogation et l'exploration des données de trajectoires construites à partir des traces GPS brutes. Il intègre un mécanisme de transformation de données, de construction de trajectoires, de stockage de trajectoires/motifs/modèles, d'interrogation de trajectoires/motifs/modèles, intègre des outils de fouille de trajectoires pour l'extraction de motifs, de construction de modèles et une interface de visualisation. Les motifs extraits et les modèles construits sont combinés afin d'améliorer la découverte de connaissances de mobilité. La découverte de ces connaissances est une interaction entre deux mondes, à savoir celui des données (trajectoires) et celui des modèles et des

[67] http://www.M-Atlas.eu/

motifs de mobilité[68] (Giannotti et al., 2011). Il y a quatre motifs de mobilité et trois modèles qui sont supportés par M-Atlas.

Les motifs représentés sur la Figure 2-34 décrivent dans l'ordre, le clustering de trajectoires (T-Cluster), les T-Patterns, les T-Flock (2.2.4.2) et le flux d'objets qui se déplacent d'une région à une autre (T-Flow).

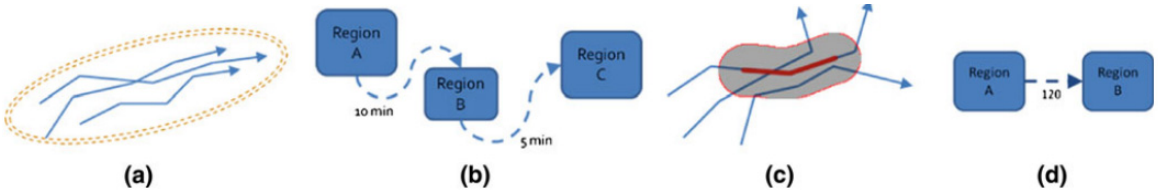



Figure 2-34 : Les motifs de mobilité supportés par M-Atlas – (a) T-Cluster, (b) T-Pattern, (c) T-Flock, (d) T-Flow (Giannotti et al., 2011).

Les modèles représentés sur la Figure 2-35 quant à eux, décrivent respectivement, l'accessibilité des partitions de trajectoires selon un seuil de distance (Accessibilité), un ensemble de motifs T-Patterns (T-PTree) et un graphe de transition labélisé par le nombre de passages entre des régions origine et destination (T-O/DMatrix).

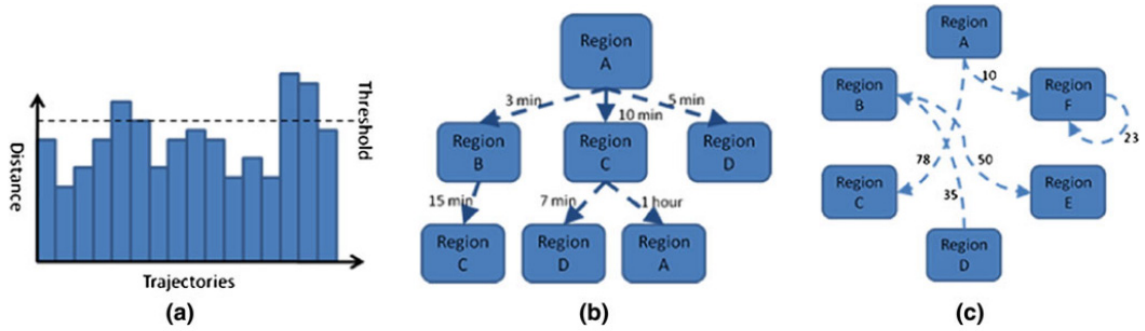



Figure 2-35 : Les modèles supportés par M-Atlas – (a) Accessibilité, (b) T-PTree, (c) T-O/DMatrix (Giannotti et al., 2011).

[68] Comportement commun d'un groupe ou un sous-groupe de trajectoires qui a été obtenu par fouille de données (Giannotti et al. 2011).

Le prototype a été développé dans l'objectif de répondre à un certain nombre de questions sur la mobilité des objets, du genre : Quels sont les motifs fréquents de déplacement ? Comment les événements influencent-t-il la mobilité ? Comment prévoir les bouchons de circulation ? Comment découvrir les embouteillages et les congestions ? etc. Il est à remarquer que les questions auxquelles répond M-Atlas sont plutôt du domaine de la fouille de données du trafic. Un exemple d'extraction de T-Patterns et du motif T-O/DMatrix est représenté sur la Figure 2-36.

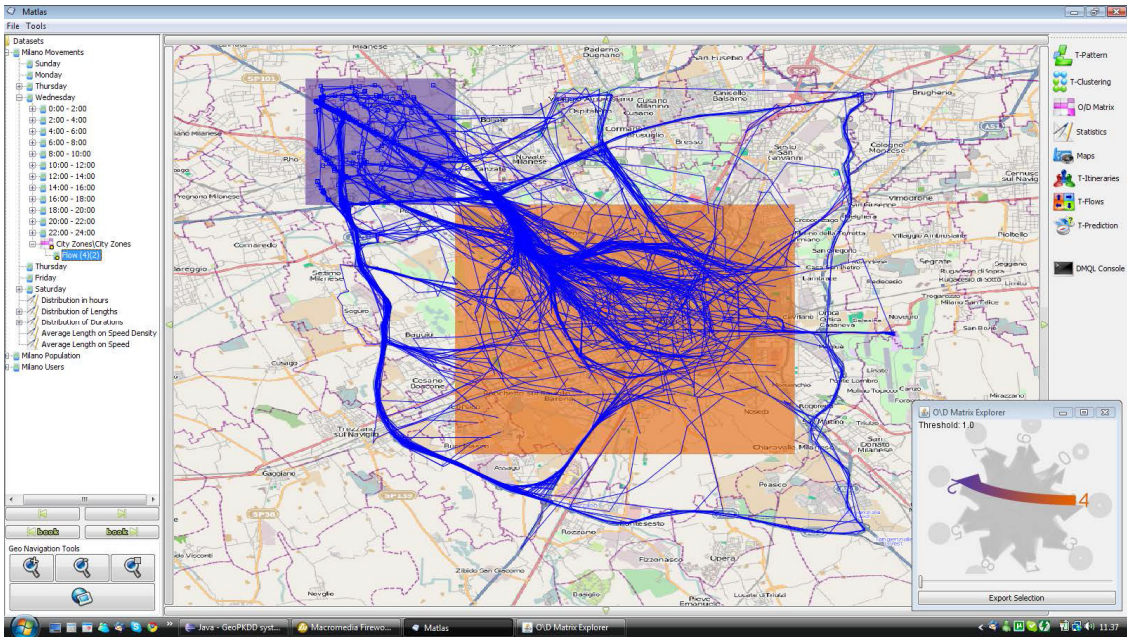



Figure 2-36 : Interface M-Atlas du résultat de T-Pattern entre le centre-ville (couleur orange) et le nord-ouest. La petite fenêtre à droite est la matrice de transition T-O/DMatrix (Giannotti et al., 2011).

## 2.4. Synthèse des méthodes de fouille de données

Nous avons exposé dans les sections précédentes un état de l'art sur des problèmes, des méthodes et algorithmes de fouille de données. Dans l'objectif de synthétiser cet état de l'art, nous présentons ci-après un tableau récapitulatif de ces méthodes et algorithmes de fouille de données :

| **Fouille de données classique** | | | |
|---|---|---|---|
| **Problèmes/Méthodes** | **Algorithmes** | **Avantages** | **Limites** |
| **Extraction de règles d'association** | Apriori | - Réduction de l'espace de recherche en exploitant une structure de données en treillis de Galois,<br>- Exploitation de l'antimonotonie du support pour élaguer le treillis,<br>- Amélioration de la sélectivité par d'autres mesures complémentaires au support et à la confiance,<br>- Simplicité de l'algorithme et efficacité pour une base de données pas très volumineuse. | - Scanne (n+1) fois la base de données /n est la taille du plus long itemset fréquent,<br>- Calcul coûteux de l'heuristique quand les itemsets fréquents sont nombreux. |
| | TreeProjection | - Utilisation d'un arbre lexicographique pour faciliter la gestion et le comptage des candidats,<br>- Limitation du comptage du support. | - Algorithme peu performant pour de grands volumes de données. |
| | FP-growth | - Nouvelle structure de données « FP-Tree » proposant un accès plus efficace,<br>- Scanne la base de données 2 fois. | - Nombre important de règles générées. |
| **Arbre de décision** | ID3 | - Algorithme de référence et l'un des plus préféré en apprentissage automatique,<br>- Non restreint aux attributs binaires. | - Ne supporte pas les variables continues et les valeurs manquantes. |
| | CART | - Précision et exhaustivité,<br>- Inférence des arbres binaires,<br>- Peut être étendu au traitement des variables continues. | - Non détection de combinaisons de variables,<br>- Besoin d'un échantillon de données de grande taille. |
| | C4.5 | - Prise en compte de variables continues,<br>- Successeur d'ID3. | - Non détection de combinaisons de variables,<br>- Introduction de variables non significatives (induisant un gain nul) dans la construction de l'arbre. |

| **Fouille de données spatiales** | | | |
|---|---|---|---|
| **Problèmes/Méthodes** | **Algorithmes** | **Avantages** | **Limites** |
| **Clustering de positions / clustering par densité :**<br>- Détecter automatiquement le nombre de clusters,<br>- Découvrir les clusters ayant des formes arbitraires. | DBSCAN | - Faible complexité O(nLogn),<br>- Résultats visuels et textuels,<br>- Validation visuelle des clusters générés. | - Rayon de voisinage fixé au préalable,<br>- Résultats pouvant variés d'une exécution à une autre à cause du choix arbitraire de la première position. |
| | OPTICS | - Unicité du résultat (le même quelque soit l'exécution),<br>- Variation du rayon de voisinage selon la densité de chaque cluster. | - Résultats visuels et non textuels,<br>- Besoin d'un index pour avoir une complexité similaire à celle de DBSCAN. |
| **Fouille de données d'objets mobiles** | | | |
| **Problèmes/Méthodes** | **Algorithmes** | **Avantages** | **Limites** |
| **Détection de trajectoires aberrantes** | TRAOD | - Complexité en $O(n^2)$/ n : le nombre de partitions de trajectoires,<br>- Découverte de partitions et de trajectoires aberrantes en se basant sur une mesure de distance et de densité,<br>- Simplification des trajectoires sans besoin de paramètres,<br>- Ne requiert pas une phase d'apprentissage automatique. | - Ne considère pas la dimension temps,<br>- Non adapté à des applications dont les objets mobiles sont contraints par un réseau. |
| **Clustering de trajectoires** | TRACLUS | - Précis et efficace,<br>- Basé sur une adaptation de l'algorithme DBSCAN. | - Sensibilité des trajectoires représentatives aux décalages des départs et arrivées des partitions de trajectoires,<br>- Partitionnement des trajectoires basé sur un algorithme approximatif (algorithme glouton). |
| **Clustering d'objets mobiles** | Flock | - Applicable directement sur des trajectoires discrètes,<br>- Prend en compte la dimension temps. | - La durée de temps des déplacements proches doit être prédéfinie d'une manière exacte,<br>- Clusters de rayon fixe. |

| | | | |
|---|---|---|---|
| | Convoy | - Temps de déplacements proches prédéfini comme un seuil minimal,<br>- Détection de clusters de formes arbitraires,<br>- Prise en compte la dimension temps,<br>- Simplification des trajectoires par une amélioration de l'algorithme Douglas-Peucker intégrant la dimension temps,<br>- Rapide et efficace, | - Besoin de transformer les trajectoires discrètes en trajectoires continues,<br>- N'intègre pas d'autres critères dans le regroupement comme la vitesse par exemple. |
| **Classement de trajectoires** | TRACLASS | - Précision de la classification à cause du regroupement basé sur les régions et les trajectoires,<br>- Utilisation des partitions de trajectoires dans la classification,<br>- Application à plusieurs domaines. | - N'intègre pas la dimension temps,<br>- Ne supporte pas encore la classification basée sur des mesures numériques. |
| **Détection des périodiques** | Mining Periodic Behaviors for Moving Objects (Periodica) | - Découverte facilitée des périodiques à cause de l'observation des déplacements à partir de zones de référence,<br>- Supporte l'incomplétude des données en entrée,<br>- Résultat non sensible au bruit (trajectoires aberrantes). | - Symbolisation des déplacements avant d'appliquer la transformée de Fourier. |

Table 2-2 : Synthèse de méthodes et algorithmes de fouille de données.

## 2.5. Conclusion

Dans ce chapitre nous avons présenté un panorama des domaines de la fouille de données qui nous intéressent essentiellement pour la suite de ce travail, à savoir, la fouille de données classique, la fouille de données spatiales et la fouille de données d'objets mobiles. Nous avons aussi posé les concepts de fouille de données (algorithmes de fouille de données, paramétrages, etc.) nécessaires à la bonne compréhension de la suite de ce travail. Pour chaque domaine de fouille de données présenté, nous avons détaillé quelques problèmes, méthodes, algorithmes et présenté des exemples d'application quand

cela était nécessaire. Nous avons également présenté deux prototypes de fouille de données à partir desquels nous nous sommes inspirés pour la conception et le développement d'un environnement d'aide à l'analyse de comportements de navires. Enfin, un tableau récapitulatif des méthodes et algorithmes de fouille de données a été présenté à la fin de ce chapitre.

Les prototypes de fouille de données présentés sont soit confidentiels, soit non adaptés pour une utilisation directe ou certains algorithmes implémentés ne répondent à aucune fonctionnalité intéressante pour notre problématique. Dans ce contexte, nous avons été amenés à concevoir et développer un atelier intégrant les méthodes de fouille de données adaptées au contexte maritime et en particulier à l'analyse des comportements à risques de navires en mer.

Dans le chapitre suivant, nous présentons la conception et le développement d'un environnement d'aide à l'analyse de comportements à risques de navires. Nous avons implémenté dans cet environnement un ensemble d'algorithmes de fouille de données adaptés pour l'extraction de connaissances sur les comportements anormaux de navires. Ces comportements anormaux peuvent être interprétés comme potentiellement à risques.