N°d'ordre NNT : 2016-LYSEI-084

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'INSA LYON

Ecole Doctorale N° 512
Mathématiques et Informatique (InfoMaths)

Spécialité de doctorat : Informatique

Soutenue le 27/09/2016, par :
Olivier Cavadenti

Contribution de la découverte de motifs à l'analyse de collections de traces unitaires

---

Devant le jury composé de :

PONCELET, Pascal Professeur des Universités Université de Montpellier Président

AZE, Jérôme Professeur des Universités Université de Montpellier Rapporteur
TERMIER, Alexandre Professeur des Universités Université Rennes 1 Rapporteur
BRUCKER, François Professeur des Universités Ecole Centrale Marseille Examinateur
MENUT, Frédéric Ingénieur de recherche ACTEMIUM Saint-Etienne Examinateur
RAISSI, Chedy Chargé de Recherche INRIA Nancy Grand-Est Examinateur
BOULICAUT, Jean-François Professeur des Universités INSA LYON Directeur de thèse
KAYTOUE, Mehdi Maître de Conférences INSA Lyon Co-Directeur de thèse

Remerciements

Cette thèse a été financée par le Projet FUI AAP 14 Tracaverre 2012-2016.

J'exprime ici ma pleine reconnaissance envers de nombreuses personnes qui m'ont aidé à mener à bien ce projet de recherche long de trois ans. Celui-ci m'a appris à quel point l'accomplissement d'une thèse CIFRE est un processus complexe et difficile et qu'il est semé de moments parfois durs et parfois joyeux.

Je remercie tout d'abord Alexandre Termier et Jérôme Azé d'avoir accepté de lire cette thèse et d'y avoir apporté leur regard et leurs remarques. Je remercie en outre François Brucker, Chedy Raïssi, Pascal Poncelet et Frédéric Menut d'avoir accepté d'être membre du jury.

Mes remerciements les plus sincères vont à mon directeur de thèse Jean-François Boulicaut pour avoir fait preuve d'une grande patience et d'un soutien sans failles. Je remercie Mehdi Kaytoue pour ses conseils et son implication précieux sans qui cette thèse n'aurait pas pu voir le jour. Je remercie Frédéric Menut et Didier Vially pour leur encadrement au sein de Courbon. Je remercie également chaleureusement Victor Codocedo pour son soutien et son aide.

J'adresse mes remerciements à mes collègues au sein de Courbon et du Liris pour leur présence et leur compagnie. Au sein de Courbon je remercie Romain, François, Ludovic, les deux Matthieu, Andrei et Sabrina. Au sein du Liris, je remercie mes collègues Guillaume, Pierre, Anès et Elise.

Pour finir, je remercie ma famille pour m'avoir soutenue dès le début, et mes amis pour leur patience et leur compréhension à mon égard. Enfin, je remercie Gabrielle pour m'avoir soutenu quotidiennement quelle que soit la situation et sans qui cette thèse n'aurait pas pu voir le jour.

Sommaire

Liste des tableaux

Table des figures

Chapitre 1

Introduction

1.1 Contextes industriel et scientifique

1.1.1 La traçabilité de produits manufacturés

Le cadre de cette thèse CIFRE (Conventions Industrielles de Formation par la RECherche) est celui de la distribution de produits manufacturés, de la production à la vente, qui est un enjeu industriel important. Ce processus est complexe, puisqu'il fait intervenir plusieurs acteurs différents, et comporte de nombreuses étapes de natures diverses. Chaque produit passe par des étapes de création, de stockage, de déplacement, de transformation (étiquetage, remplissage de produits), de vente et de destruction. Chaque étape peut être effectuée dans des endroits différents (usines, entrepôts, véhicules, magasins) appartenant à divers acteurs de la chaîne logistique entrainant des types de comportements distincts. Dans de nombreux domaines, comme la traçabilité de médicaments, ou l'agro-alimentaire, la traçabilité unitaire de produit permet d'avoir une vision d'ensemble des processus métiers effectués. Tracer les médicaments par exemple, permet d'observer les flux de produits dans différentes régions du globe, et d'analyser si ces produits passent par des zones dites à risque (régions ou pays non prévus pour la livraison d'un certain type de médicaments). En effet, la notion de marché est importante, et les produits fabriqués à un endroit sont généralement destinés, par contrat, à certains pays bien précis. Effectuer la traçabilité de denrées alimentaires ou de médicaments, permet aussi de réduire les risques en ayant accès à la provenance de ceux-ci et ainsi on s'assure de leur qualité. La Figure 1.1 montre le processus global de la traçabilité unitaire. Chaque produit est acheminé par des acteurs de la chaîne logistique reliés par des flux physiques, c'est-à-dire décrivant les transports du produit d'une localisation à l'autre. Chaque acteur possède un système de traçabilité interne aux sites qu'il gère (par exemple le réseau des bases de données de ses usines) et chacun peut récupérer les ensembles d'évènements liés à la traçabilité des produits sur les sites le concernant. Le nombre total de sites peut varier suivant la taille du réseau manufacturier mais dans notre projet les entreprises qui conçoivent les produits possèdent entre trois à quatre sites de productions, une dizaine d'entrepôts et de distributeurs. Le nombre de magasins et clients finaux est en revanche

non connu. Ainsi, on peut considérer que le réseau sera constitué de plusieurs centaines de sites dont certains comme les magasins ne fourniront sans doute pas ou peu de données. Les sites appartenant aux entreprises productrices sont en faible nombre mais fourniront quand à eux plus de données. Ainsi, ces informations permettent aux agents logistiques d'accéder aux données concernant leurs processus métiers et d'accompagner le travail des ingénieurs de contrôle qualité ou les analystes de la traçabilité. Dans un second temps, la traçabilité externe permet dans l'idéal d'effectuer la traçabilité des produits lorsqu'ils quittent les sites (traçabilité sur les routes, les aéroports, les chemins de fer, etc). Cette traçabilité permet théoriquement de reconstituer l'ensemble de cycle de vie d'un produit grâce à son identifiant EPC (Electronic Product Code) et en interrogeant chaque base de données internes et externes aux sites industriels. L'objectif final peut être d'établir une architecture qui permette d'obtenir la traçabilité totale d'un produit à partir de son identifiant et ainsi accéder à l'ensemble de l'historique des actions effectuées sur lui. Une étape permettant d'agréger les différentes sources de données est alors primordiale. Récupérer l'ensemble de ces informations permet aux agents logistiques participant à la création initiale des produits de suivre le cycle de vie global des produits. En effet, ces agents signent des contrats pour l'acheminent des produits à des destinations précises, prenant compte des contextes économiques de chaque pays. La vente des produits dans d'autres pays que ceux fixés par contrat est illégale, et consiste en un marché gris, et est un enjeu important pour les producteurs. Ces objectifs sont le cas idéal de fonctionnement d'une offre de traçabilité unitaire. Or, on constate que plusieurs contraintes physiques et logistiques ne permettent pas d'effectuer entièrement cette traçabilité. L'absence de capteurs, la confidentialité de certaines données, des accidents provoquant des défaillances de matériel ou des erreurs de livraisons entraînent une vision partielle de la traçabilité des produits. Dans la pratique les conditions de traçabilité peuvent donc entrainer des problèmes de reconstitution des traces unitaires. De la même façon, des facteurs humains peuvent compromettre le bon fonctionnement de l'acheminement des produits. En effet, si des dysfonctionnements apparaissent (livraisons non effectuées, détournements de produits pour les copier, accidents amenant la destruction de produits, etc.), la perte financière peut être très importante (perte de conteneurs de marchandises, disparition de produits qui possèdent des valeurs élevés, comme les produits de luxe ou le matériel informatique. Prévenir et limiter ces dysfonctionnements sont alors des enjeux capitaux pour le milieu industriel, notamment en ayant des informations sur les contextes de ces incidents. Par exemple, connaître les lieux ou les périodes de temps durant lesquels sont survenus ces incidents peut permettre d'effectuer des analyses sur les produits et acteurs concernés.

Depuis une quinzaine d'années, les puces RFID (de l'anglais Radio Frequency IDentification) ou les systèmes de code barres comme les QRCode sont devenus courants. Ces systèmes permettent d'identifier rapidement un objet ou un individu grâce à des informations embarquées. De nombreuses applications existent : les accès aux transports publiques, les péages, le suivi de produits au sein des usines, l'identification de conteneurs de médicaments ou de denrées alimentaires, etc. EPCglobal est une organisation qui travaille à proposer un standard international en vue de normaliser les usages des systèmes

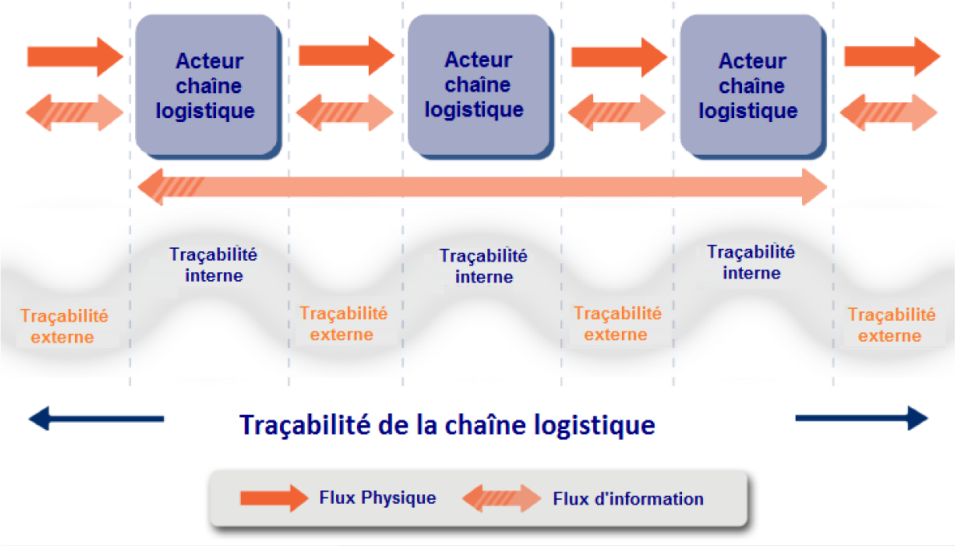


FIGURE 1.1 – Processus global de la traçabilité de produits le long de la chaîne logistique

d'identification de produits grâce à un protocole nommé EPCIS (Electronic Product Code Information System [1]). La traçabilité de produits étant unitaire et le nombre de produits potentiellement très élevé (plusieurs centaines de millions par an), les agents logistiques se trouvent alors face à une masse très importante de données de traçabilité. En effet, le nombre d'enregistrements par an pourra avoisiner de un à plusieurs milliards, ce qui posera des problèmes tant au niveau stockage que de la transformation pour l'analyse de données.

Dans notre cas d'étude, les données de traçabilité sont structurées selon le standard EPCIS. A chaque produit est assigné une séquence de plusieurs évènements EPCIS et c'est cette séquence que l'on appelle une trace unitaire. Elle contient alors l'ensemble des données de traçabilité captées. Ces données captées peuvent diverger des données que l'on a supposées a priori captables et attendues par le système manufacturier et le réseau logistique mis en place. Cette divergence sera utile pour servir de support à l'extraction d'anomalies sous la forme d'ensembles d'évènements, mais aussi à la compréhension du réseau. De fait, nous voyons le processus d'analyse des traces unitaires comme un exemple typique de processus d'extraction de connaissances à partir de données pour lequel il faut mobiliser des méthodes de fouille de données.

Le domaine industriel est porteur de nombreuses connaissances variées tant sur le système manufacturier, les traces unitaires de produits, ou la nature des informations que l'on souhaite extraire de ces données. Les analystes des acteurs propriétaires des produits et des données de traçabilité, possèdent une connaissance du système manufacturier, ses localisations et les objectifs de chacune, une connaissance des produits (contenu, forme, prix,...), une connaissance des intérêts et des objectifs des acteurs logistiques et

1. http://www.gs1.org/sites/default/files/docs/epc/epcis_1_1-standard-20140520.pdf

une connaissance du domaine (temps de stockage, distances, temps de livraisons) qui est cependant partielle et propre à chaque acteur. Il est alors difficile d'avoir une vision globale et encore plus de détecter des anomalies. Ces données procurent des informations diverses telles que les positions dans le temps et dans l'espace, la nature des actions faites sur le produit, l'identité de ceux qui agissent sur le produit, etc... Des méthodes classiques existent pour permettre de collecter, stocker et interroger ces données de traçabilité stockées dans des bases de données relationnelles. Des techniques axées fouille de données se sont attachées à extraire des phénomènes précis et adaptés aux données de traçabilité comme décrire les circuits de distributions, les similarités de comportements, les comportements frauduleux ou fréquents, les évènements manquants, etc...

Interroger les traces unitaires de façon classique, via des requêtes de type SQL, n'est pas suffisant lorsqu'il s'agit de découvrir des connaissances jusqu'ici ignorées des analystes du domaine manufacturier. En effet, si les analystes peuvent connaître les produits détournés ou concernés par des accidents, ils leur est impossible de connaître les contextes durant lesquels arrivent ces anomalies, étant donné le nombre de possibilités très importante. Par exemple, si un contexte est modélisé par une ensemble de n propriétés booléennes (par exemple, "le produit est passé par l'usine A"), alors le nombre de combinaisons totales est de 2^n. Pour explorer ces combinaisons, on peut s'intéresser aux méthodes de découverte de motifs bien étudiées pour le processus d'extraction de connaissance à partir de données (ECD).

1.1.2 L'extraction de connaissances à partir de données

Un processus d'Extraction de Connaissances à partir de Données (ECD) est constitué de quatre étapes permettant d'analyser des données et d'extraire de la connaissance sans avoir, en amont, d'hypothèses précises sur celles-ci. La sélection à partir des données brutes permet de choisir un sous-ensemble de données en adéquation avec ce que l'on veut étudier. L'analyste peut sélectionner uniquement certaines traces de produits (répondant à des critères de taille, poids, gamme,...) ayant été vendus sur certains marchés durant une période de temps donnée. Durant cette étape, il est possible de traiter ces données qui peuvent contenir du bruit et des données manquantes à cause de contraintes physiques qui peuvent influer sur l'enregistrement des données : données aberrantes, données non captées, problèmes de capteurs ou de stockage. Les traces unitaires possèderont généralement des évènements manquants voir des séquences d'évènements non présentes (si un site entier ne fait pas parvenir ses données lors de l'agrégation des données de traçabilité). Finalement, même si les données agrégées sont présentes, il est également possible que celles-ci soient non valides selon certains critères fixés par les analystes des producteurs de produits (les évènements provenant de distributeurs ou vendeurs non autorisés par contrat par exemple).

A partir des données sélectionnées, une étape de transformation est effectuée pour les rendre exploitables par des algorithmes de fouille de données. On peut construire un modèle prédictif qui va permettre, à partir d'exemples étiquetés par une classe, de déduire la classe des exemples futurs. Nous montrerons que si l'on a une base de traces comportementales étiquetées avec l'identité de l'agent, il est possible dans certains cas

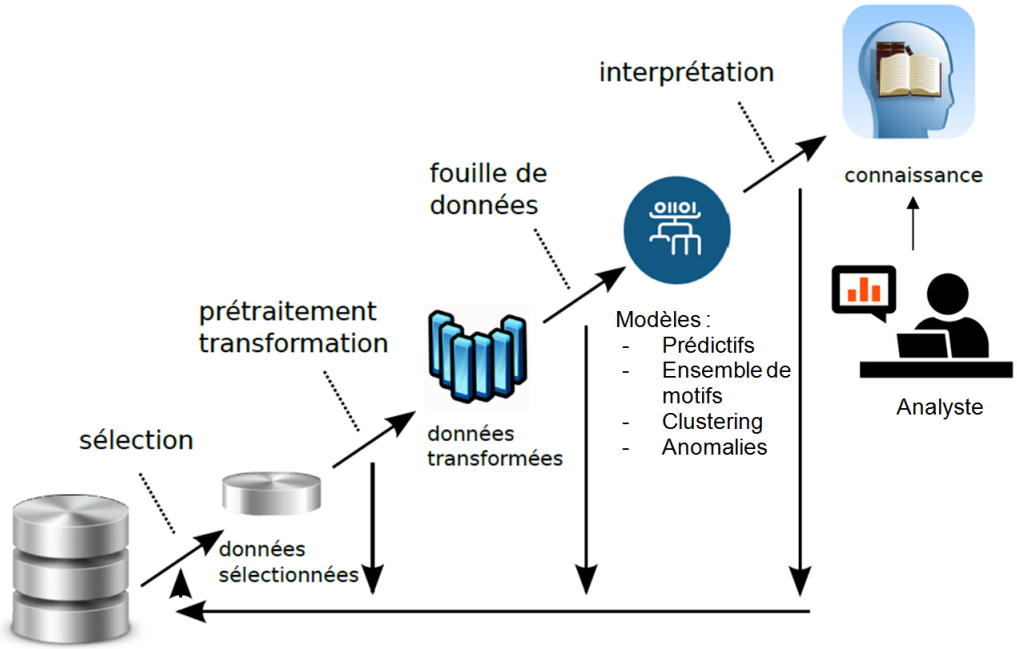


FIGURE 1.2 – Extraction de Connaissances à partir de Données (ECD)

d'application, de déduire l'identité de cet agent à partir des traces. Un autre champ de la fouille de données est la classification automatique, qui permet de construire des groupes d'objets qui vont minimiser une fonction donnée. Un des algorithmes le plus connu est K-means (MacQueen [64]) qui divise les objets en un nombre fixe de groupes, appelés clusters, en considérant la distance d'un point à la moyenne des points du groupe auquel il appartient. La découverte de motifs est un champ de la fouille de données qui s'attache, à proposer un ensemble de motifs caractérisants les données. Le calcul de motifs caractérisant des groupes d'objets via des ensembles de propriétés qu'ils partagent en commun a été particulièrement étudié. L'une des premières applications phares en découverte de motifs concerne les analyses d'achats dans les magasins, avec l'exemple de l'extraction des ensembles de produits souvent achetés ensembles par Agrawal et al. [3]. Dans la Table 1.1 nous proposons un exemple simple de quatre produits décrits par quatre propriétés booléennes : une croix dans une case exprime le fait qu'un produit de la ligne vérifie la propriété correspondant à la colonne. Le produit p_1 possède deux propriétés s_1 et s_2 ce qui signifie par exemple qu'il est passé par les sites 1 et 2. Pour caractériser des données ensemblistes, la fouille de motifs permet de regrouper les ensembles d'objets qui vont partager un même ensemble de propriétés. On peut observer que l'ensemble de tous les produits possèdent la propriété s_1. De même, les produits p_1 et p_2 ont les propriétés s_1 et s_2 en commun. Depuis, de nombreux langages pour exprimer les motifs ont été proposés comme les séquences d'ensembles ([73]) permettant d'étudier l'ordre des évènements, ou encore de prendre en compte le temps et la distance spatiale entre les objets [33] sous la forme de trajectoires spatio-temporelles. Nous nous intéressons dans ce manuscrit au

cas particulier des données ensemblistes que l'on construira à partir des traces unitaires grâce à un procédé de transformation adapté et un encodage de la connaissance experte. Plus particulièrement, nous nous inscrirons dans le champ de la fouille de motifs caractéristiques d'anomalies grâce des données ensemblistes disposant de classes binaires. Nous reprenons l'exemple de la Table 1.1 en assignant à chaque instance une classe + (positive) ou une classe − (négative) dans la Table 1.2. Nous pouvons observer que les deux produits p_3 et p_4 sont assignés à la classe négative, et possèdent tous les deux un ensemble de propriétés $\{s_1, s_3, s_4\}$ qui n'est pas présent parmi les produits possédant la classe positive. Selon les méthodes d'assignation de la classe aux produits, ainsi que la connaissance exprimée via les propriétés booléennes, il est alors possible d'extraire des ensembles de propriétés caractéristiques des collections de traces négatives. Nous proposons alors une méthode pour assigner des classes positives ou négatives en fonction d'un modèle, sous forme de graphes, en s'appuyant sur des données comportementales.

	s_1	s_2	s_3	s_4
p_1	×	×		
p_2	×	×	×	
p_3	×		×	×
p_4	×		×	×

TABLE 1.1 – Exemple de données ensemblistes

	s_1	s_2	s_3	s_4	$classe$
p_1	×	×			+
p_2	×	×	×		+
p_3	×		×	×	-
p_4	×		×	×	-

TABLE 1.2 – Exemple de données avec une classe assignée

Dans la dernière étape, les motifs sont restitués visuellement à l'analyste pour une phase d'interprétation de la pertinence. On peut mobiliser différentes contraintes préétablies (fréquence d'apparition de certaines propriétés dans les données par exemple). Si nous reprenons l'exemple de la Table 1.1, nous pouvons nous intéresser uniquement aux motifs ensemblistes qui vont décrire au minimum trois produits. Ainsi, l'ensemble $\{s_1, s_2\}$ ne sera pas extrait, mais les ensembles $\{s1\}$ et $\{s_1, s_3\}$ le seront, ce qui réduit le nombre de résultats calculés et montrés. Il est possible pour l'analyste de reprendre le processus de fouille de données à chaque étape suivant les résultats que l'on fournit, que ce soit au niveau de la sélection, la transformation ou l'extraction. L'analyste peut choisir de modifier les critères de sélection des traces unitaires (par exemple, s'intéresser à un autre marché, ou à d'autres gammes de produits), modifier la connaissance exprimée par les propriétés, en construire des nouvelles ou encore modifier les contraintes d'extraction des motifs (par exemple, modifier le seuil de fréquence).

Actuellement, la valorisation des traces unitaires se fait via l'interrogation classique

sous la forme de tableau de bord, ou par la découverte d'évènements bien décrits a priori comme les retours de produits, ou les successions d'évènements non valides [53] [78] au regard de règles métiers, mais sans exploration des données proprement dites. Nous proposons un cadre méthodologique pour permettre de guider les analystes en vue de construire des scénarios à partir des traces et d'y extraire de multiples informations comme des comportements et évènements fréquents ou encore des ensembles de propriétés qui décrivent des comportements jugés anormaux au regard d'une connaissance experte pré-établie.

1.2 Problèmes et contributions

Avec la disponibilité de gros volumes de collections de traces captées, les analystes cherchent à mieux connaître leurs systèmes d'informations grâce à la proposition de méthodes permettant la collecte et l'analyse de ces traces. Ces traces unitaires obéissent à des contraintes strictes, tributaires des processus métiers, et sont structurées sous la forme de séquences d'évènements spatio-temporels pouvant être enrichies d'informations issues de la connaissance du domaine. Ces données sont issues des capteurs présents sur les sites industriels et possèdent une structure propre au milieu d'application et au standard adopté, dans notre cas le protocole EPCIS. Une trace unitaire est issue d'un processus d'agrégation des évènements EPCIS correspondant à un identifiant unique. Il s'agit alors de récupérer l'ensemble des évènements dans chaque base de données des sites industriels puis de les reconstituer sous forme d'une séquence d'évènements ordonnés dans le temps. Or, ces données seront rarement, voir jamais, complètes ce qui fait que les traces unitaires sont le résultat de ce qui est possible de capter avec l'état actuel du réseau. L'ensemble de ces données captables peut être exprimé sous la forme d'une structure unique que l'on nomme *modèle de filière*. Celle-ci exprime une vérité terrain, c'est-à-dire une connaissance de référence sur le fonctionnement attendu du système manufacturier en fonctionnement, comme par exemple les séquences de sites par lesquels doivent passer les produits. Dans le cas de ce FUI (Fonds Unique Interministériel), les données réelles de traçabilité ne sont pas disponibles car les chaînes logistiques ne sont pas fonctionnelles. En revanche nous connaissons leur structure et comment ces données seront récupérées. Nous avons choisi dans ce manuscrit de travailler sur des exemples que l'on pourra rencontrer une fois le système établi. Pour une de nos méthodes, nous avons utilisé un jeu vidéo de gestion de réseau manufacturier pour la valider. Le premier objectif est de fournir un cadre méthodologique qui permettra de traiter l'ensemble des processus de gestion des traces unitaires, de la sélection à la transformation jusqu'à l'analyse des données traitées (les contextes de fouille). L'étape cruciale est le codage intelligent de propriétés booléennes pour permettre, par la suite, d'extraire de l'information ayant du sens pour les analystes. En effet, un mauvais encodage de propriétés ne permettra pas d'extraire d'informations pertinentes au regard des problématiques de la fouille de traces (extraction de comportements fréquents ou anormaux). Dans une deuxième contribution, nous nous focalisons sur la découverte de comportements anormaux dans les traces unitaires en confrontant les deux visions des données captées et

captables. Nous nous servons dans un premier temps du modèle de filière pour étiqueter chaque trace comme étant normale ou anormale par rapport à ce modèle (en analysant l'absence des transitions entre les sites). Dans un deuxième temps, nous appliquons, sur un contexte de fouille donné, une méthode de fouille de motifs discriminants qui vont caractériser la collection de traces anormales. Enfin, notre troisième contribution s'attaque à la problématique de l'usurpation d'identités d'un agent. Nous démontrons en premier lieu que l'intuition selon laquelle on peut détecter un agent à partir de sa collection de traces est juste, avec un premier cas d'application. Quand un agent possède plusieurs identités, un modèle prédictif hésitera lors du processus de classification entre ces identités. Nous nous servons de cette observation comme base pour une méthode s'appuyant sur l'analyse de concepts formels et notamment sur les structures de patrons [31] qui va extraire les identités que l'on soupçonnent appartenir à un même agent.

1.2.1 Un cadre méthodologique pour la fouille de données appliquée aux traces unitaires

Le contexte de la traçabilité de produits manufacturés implique la disponibilité de grandes quantités de traces unitaires possédant de nombreux attributs différents (temps, positions, type de produits, type d'actions). Notre première contribution consiste à rendre disponible un cadre méthodologique, validé par un logiciel, permettant de gérer, traiter et analyser ces traces de multiples façons. Nous proposons d'adapter les étapes du processus d'extraction de connaissance à partir de données à la fouille de traces unitaires. Premièrement, le cadre méthodologique propose de sélectionner les traces unitaires en accord avec des interrogations classiques de l'analyse de traces unitaires, comme par exemple la sélection de traces unitaires appartenant à des ensembles de produits issus d'un marché spécifique (Asie, Europe) ou d'un type précis (marque ou type de médicament). Ensuite, une fonction du cadre méthodologique est de produire des propriétés décrivant une connaissance plutôt élémentaire présente dans les traces de manière implicite. La construction de ces propriétés pose un défi tant au niveau volumétrique concernant le calcul même des propriétés, qu'au niveau sémantique, c'est-à-dire produire des propriétés utiles à la caractérisation de connaissances intéressantes en fonction de l'objectif voulu. La construction de ces scénarios permet, à partir d'une collection de traces donnée, d'une part de répondre à des interrogations variées en sélectionnant les traces possédant un ensemble de propriétés, d'autre part en appliquant des algorithmes connus et efficaces en fouille de motifs.

	loc_A	*loc_B*	*loc_C*	*vente_janvier*	*vente_février*	*vente_mars*	*vendu*
1	×	×	×		×		×
2	×	×					
3	×	×	×			×	×

TABLE 1.3 – Exemple d'un contexte de fouille simple à partir de traces unitaires

Nous exposerons dans ce manuscrit des cas de scénarios traitant des données sous la

forme de tables binaires, domaine de motifs très bien étudié, composées de propriétés booléennes conçues grâce au résultat de l'évaluation d'une expression ou d'un prédicat. Chaque ligne correspondra à une trace unitaire, et chaque colonne portera une croix si la trace correspondante vérifie cette propriété ou non. Par exemple, la Table 1.3 montre un contexte de fouille simple issue des évènements de traces unitaires de la Table 1.4. Cette table expose trois traces différentes ayant un identifiant d'évènement *ev_id*, un identifiant de produits *epc_id*, le parent d'un produit *epc_parent*, une estampille temporelle qui indique quand l'événement a été capté, le type de l'évènement *biz_step* et la localisation *biz_loc*. Les *epc_id* numéro 4 et 5 correspondent à des palettes et sont les parents de nos trois produits. Pour reconstituer les traces unitaires de chaque produit, on considère les évènements des palettes comme faisant partie des produits qu'elles contiennent. Par exemple, le produit 1 est passé par les localisations loc_A, loc_B et loc_C et a été vendu durant le mois de février. Rapidement, on peut voir que tous produits sont passés par les localisations loc_A et loc_B, et seuls les produits 1 et 3 sont en plus passés par loc_C et ont été vendus, le deuxième n'étant pas encore vendu. Nous exposerons une série de transformations plus avancées dans le Chapitre 2.

Notre cadre méthodologique propose également d'utiliser la connaissance experte du domaine manufacturier de différentes façons, sous la forme du modèle de filière. Nous décrirons en détail comment construire, mettre à jour, et visualiser ce modèle de filière, permettant de généraliser les comportements observés du domaine de la traçabilité de produits. En effet, celui ci peut être construit manuellement, ce que nous proposons à l'analyste via une interface du logiciel, ou être dérivé d'une collection de traces unitaires issue d'une requête. La construction, l'évolution, la visualisation du modèle filière ainsi que son intégration dans le processus de fouille posent des problématiques centrales pour la valorisation de traces puisqu'il pourra interagir avec les différents concepts (traces, traces transformées, motifs).

Pour démontrer la pertinence du cadre méthodologique, nous avons mis en place une plateforme web. Celle ci est composée d'une interface web permettant d'afficher les différentes étapes du processus de fouille de traces, sélection, transformation, fouille du contexte, affichage des motifs, et visualisation du modèle de filière. Cette interface utilise notre algorithme de fouille de motifs sur le contexte de fouille conçu par l'utilisateur, avant d'afficher les ensembles de propriétés ainsi que les chemins parcourus par le groupe de produits correspondant grâce au modèle de filière.

1.2.2 Découverte de motifs intelligibles et caractéristiques d'anomalies dans les traces unitaires

Le volume de données disponible permet d'avoir une certaine vision du système en fonctionnement, qui est plus ou moins partielle, suivant le degré de complétude des données disponibles. Les systèmes manufacturiers ont des processus métiers bien appliqués par les agents qui opèrent des transformations sur les produits, comme la conception, les déplacements ou la vente des produits. Il est possible, à partir d'un ensemble de traces, d'avoir une vision globale des comportements fréquents, en produisant une généralisation des données par exemple sous la forme d'un graphe des flux d'objets, que nous avons

ev_id	epc_id	epc_parent	time	biz_step	biz_loc
1	1	-1	2016-01-01 05 :11 :33	COMMISSIONNING	loc_A
2	2	-1	2016-01-01 05 :11 :33	COMMISSIONNING	loc_A
3	3	-1	2016-01-01 05 :11 :33	COMMISSIONNING	loc_A
4	4	-1	2016-01-01 05 :11 :33	COMMISSIONNING	loc_A
5	1	4	2016-01-01 05 :25 :20	PACKING	loc_A
6	2	4	2016-01-01 05 :25 :20	PACKING	loc_A
7	3	4	2016-01-01 05 :25 :20	PACKING	loc_A
8	4	-1	2016-01-01 05 :25 :20	PACKING	loc_A
9	4	-1	2016-01-01 05 :36 :45	STORING	loc_A
10	4	-1	2016-01-02 13 :05 :10	SHIPPING	loc_A
11	4	-1	2016-01-04 16 :25 :46	RECEIVING	loc_B
12	4	-1	2016-01-04 16 :25 :46	UNPACKING	loc_B
13	4	-1	2016-01-04 16 :25 :46	DECOMMISSIONNING	loc_B
14	1	4	2016-01-04 16 :37 :20	FILLING	loc_B
15	2	4	2016-01-04 16 :37 :20	FILLING	loc_B
16	3	4	2016-01-04 16 :37 :20	FILLING	loc_B
17	5	-1	2016-01-04 16 :47 :15	COMMISSIONNING	loc_B
18	1	5	2016-01-04 16 :47 :15	PACKING	loc_B
19	2	5	2016-01-04 16 :47 :15	PACKING	loc_B
20	3	5	2016-01-04 16 :47 :15	PACKING	loc_B
21	5	-1	2016-01-04 16 :47 :15	PACKING	loc_B
22	4	-1	2016-01-04 16 :47 :15	STORING	loc_B
23	5	-1	2016-01-05 06 :12 :33	SHIPPING	loc_B
24	5	-1	2016-01-08 13 :22 :01	RECEIVING	loc_C
25	5	-1	2016-01-08 13 :22 :01	UNPACKING	loc_C
26	5	-1	2016-01-08 13 :22 :01	DECOMMISSIONNING	loc_C
27	1	-1	2016-02-12 11 :01 :25	RETAIL_SELLING	loc_C
28	3	-1	2016-03-24 18 :32 :01	RETAIL_SELLING	loc_C

TABLE 1.4 – Un ensemble de traces unitaires

introduit précédemment sous le nom de modèle de filière. Celle-ci peut prendre la forme d'un graphe orienté : les différentes localisations composant le système manufacturier (usines, entrepôts, magasins) sont modélisées par des nœuds, et les arcs désignent des flux d'objets attendus par les industriels entre deux localisations. Or les données peuvent ne pas correspondre à ces données captables : des produits peuvent être détournés et ne pas être reçus sur un autre site (ou un site non prévu) ce qui peut être synonyme d'un oubli de scan ou bien d'un vol. Plus subtilement, ces produits peuvent réapparaître plus tard dans le circuit de distribution mais via un chemin non souhaitable. La Figure 1.3 montre, à gauche, un modèle de filière comportant dix sites et deux exemples de détournement de produits. La figure du centre montre, en rouge, les localisations successives captées d'un ensemble de produit partants de A pour parcourir les localisations C, D, E et J. Comme on le voit la transition entre E et J n'est pas présente dans le modèle de filière décrit par la figure à gauche. De la même façon, la figure de droite montre des produits allant de B vers C, G, I puis revenant à C. Également, la transition entre I et C n'est pas présente dans le modèle de filière initial. Nous proposons d'utiliser cette différence, le nombre de transitions non conformes (en rouge), entre les données captées, dont nous aurons extrait les informations spatiales, et les données captables présentes via le modèle de filière pour labelliser nos traces, transformées en contextes de fouille, avec une classe $+$ ou $-$. Ces exemples font tout à fait sens dans le cas industriel précis

considéré dans le cadre de cette thèse.

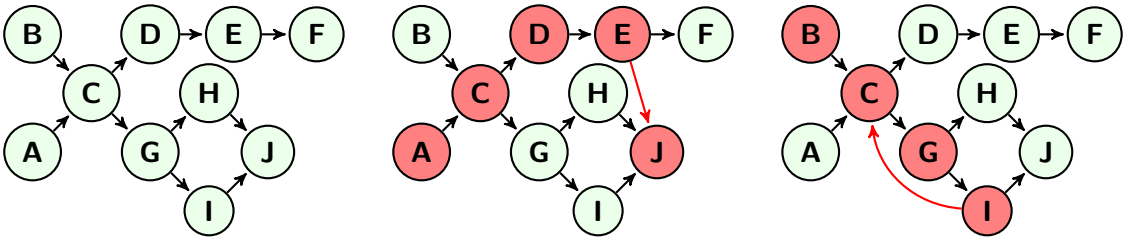


FIGURE 1.3 – Un modèle de filière ainsi que deux exemples de détournements anormaux

Un des moyens de détecter ces comportements anormaux est de fixer en amont les flux d'objets, les paires de localisation, que les experts considèrent comme autorisées pour classer les traces en deux catégories, d'une part celles qui possèdent une proportion élevée de transitions classées comme non autorisées et d'autre part, celles avec moins de transitions non autorisées classées comme normales. La connaissance servira aussi, comme dans le cas des anomalies contextuelles, à décrire les collections de traces suspectées anormales.

Nous avons vu que nous pouvons décrire nos traces unitaires sous la forme de contexte de fouille, connus dans la littérature comme des données ensemblistes (ou tables binaires). Pour extraire des co-occurences de propriétés booléennes (des motifs), sans énumérer l'ensemble des possibilités combinatoires, de nombreuses contraintes ont été proposées. Chaque contrainte cible un objectif précis, en fonction de la sémantique des motifs, comme par exemple la contrainte de fréquence pour extraire les ensembles récurrents ou rares dans les données, ou la définition de contraintes basées sur des mesures statistiques. Un champ de la fouille de motifs s'est intéressé à l'extraction de motifs en se basant sur des modèles mathématiques [29], des ensembles de règles [4] ou encore des réseaux bayésiens [46]. L'inconvénient de ces méthodes est que la production d'une connaissance experte pertinente peut être difficile et longue. Il faut alors définir des outils spécifiques pour produire de tels modèles, ou alors les construire automatiquement. Il est aussi important de proposer des modèles évolutifs suivant les résultats des algorithmes de fouille, c'est-à-dire la possibilité de corriger le modèle si l'expert juge qu'un motif considéré comme pertinent ne l'est finalement pas. Notre méthode combine alors l'utilisation de modèles experts, le modèle de filière, pour classifier les traces unitaires transformées en contexte de fouille spécialement conçu pour l'extraction de motifs, et ainsi extraire les motifs caractéristiques de collections de traces anormales, au regard du modèle de filière, en utilisant les motifs émergents (ou discriminants/corrélés) [25].

Dans le cas de la traçabilité de produits manufacturés, la fouille de traces s'appuie sur des données plus ou moins confidentielles. En effet, chaque entreprise possède une base de données différente pour l'enregistrement de ces données de traçabilité. La confidentialité des données nous a poussé à trouver un autre cas d'application réel, qui fournit de nombreuses traces de comportement, qui est celui des traces de jeux vidéo. Ce domaine nous intéresse pour deux points majeurs présents également dans le domaine des traces de produits : d'une part la présence d'une connaissance experte forte qui permet de

présupposer des comportements voulus ou attendus via les règles même du jeu et ses stratégies pour réussir l'objectif principal du jeu ; et d'autre part la présence de traces de comportements (sous la forme de séquences d'évènements) produites par des sources (ici les joueurs). On observe de la même manière les données captables, c'est-à-dire les données qui peuvent être générées ou qui sont attendues durant une partie, et les données captées, qui correspondent aux actions des joueurs.

Pour la validation de notre méthode d'extraction de motifs caractéristiques d'anomalies nous avons choisi un premier jeu générant des traces synthétiques issues de déplacement de véhicules de fret de OpenTTD [2]. Nous avons construit des scénarios synthétiques pour établir la proportion de produits ayant des comportements anormaux qui sont décrits par les motifs, puis nous avons étudié le passage à l'échelle de notre méthode en terme de temps d'exécution. Concernant les données réelles, nous avons utilisé celles du jeu vidéo Defense of the Ancients 2 [3]. Nous avons appliqué notre méthode de description des collections de traces anormales pour extraire l'ensemble des mouvements courants des personnages sur la carte de jeu puis ainsi détecter les mouvements qui sont non conformes à la norme et les décrire via des propriétés ce qui nous a permis d'extraire des motifs caractéristiques d'anomalies (stratégies erronées).

1.2.3 Identification d'agents usurpant une identité

Cette contribution s'attaque au problème de l'usurpation d'identités par des acteurs de la chaine logistique. Par exemple un distributeur qui est banni de la chaine logistique pour cause de fraude peut revenir au sein de celle ci sous une autre identité. Nous faisons l'hypothèse qu'à partir de l'encodage de descripteurs spécifiques à partir des traces unitaires, et de l'identité de l'agent, nous pouvons construire un modèle prédictif permettant de déduire l'identité d'un agent à partir d'un ensemble de traces comportementales de manière efficace. La Table 1.4 est une transformation des traces à partir du contexte de fouille, présenté dans la Table 1.3, en traces comportementales avec des attributs numériques (on considère dans cet exemple que l'on a deux traces unitaires par agents). Le premier attribut indique le nombre de localisations traversées par le produit, les deuxième et troisième indiquent quand a lieu la vente (en février ou mars) et on labellise avec le nom de l'agent (ici on s'intéresse aux ventes et l'on considère que l'on assigne le nom des vendeurs).

On observe que l'on peut bien séparer les trois agents puisqu'ils possèdent des valeurs caractéristiques dans chaque dimension. Dans ce cas précis, le modèle prédictif reconnaît parfaitement les agents, et on obtient alors une matrice de confusion telle que présentée dans la table de gauche de la Figure 1.5, où toutes les valeurs de la diagonale sont égales à 1 (le classifieur a reconnu toutes les traces d'un agent comme étant bien les traces de cet agent). Nous avons démontré avec notre troisième contribution que les agents pouvaient être efficacement reconnus à partir de ce type de traces de comportements dans le cadre précis du jeu vidéo. Or, dans le cas où deux agents sont en réalité un même

2. `https://fr.wikipedia.org/wiki/OpenTTD`
3. `https://fr.wikipedia.org/wiki/Dota_2`

vendeur	*nombre_locs*	*vente_février*	*vente_mars*
a_1	3	1	0
a_1	3	1	0
a_2	2	0	0
a_2	2	0	0
a_3	3	0	1
a_3	3	0	1

FIGURE 1.4 – Traces comportementales

agent, alors le classifieur va distribuer les ensembles de traces parmi ces deux agents. La table de droite de la Figure 1.5 montre une autre matrice de confusion dans laquelle la moitié des traces de a_2 et a_3 sont classifiées comme étant les traces de l'autre agent. Si cela ne permet plus de reconnaître chaque agent individuellement à partir de ses traces, cela rend possible en revanche l'identification les ensembles d'agents qui sont susceptibles d'être un unique agent sous plusieurs identités. Il s'agit alors d'extraire cette distribution entre deux agents, ce que nous faisons via une méthode d'analyse de concepts formels [30] et plus particulièrement grâce aux structures de patrons [31].

	a_1	a_2	a_3
a_1	1	0	0
a_2	0	1	0
a_3	0	0	1

	a_1	a_2	a_3
a_1	1	0	0
a_2	0	0.5	0.5
a_3	0	0.5	0.5

FIGURE 1.5 – Exemples simples de deux matrices de confusion

Nous avons appliqué cette méthode de fouille de traces sur le jeu Starcraft 2. Notre acteur logistique correspond ici à un joueur qui effectue des parties, sous une identité que l'on appellera avatar, et qui possède des traces comportementales de jeu (séquences d'actions stratégiques). Dans certains cas, un même joueur incarne plusieurs avatars virtuels différents (cas de multi-comptes). Un joueur professionnel peut ainsi cacher son niveau pour s'entrainer, ou alors gagner des compétitions en ligne de plus faible niveau sans être reconnu, ce qui peut poser des problèmes dans le cas de tournois de qualification majeurs mettant en jeu de l'argent. En effet, le domaine de l'e-sport est un phénomène sociétal en pleine croissance [87].

1.3 Organisation du manuscrit

Le premier chapitre de ce manuscrit développe un état de l'art de plusieurs domaines, importants à différents niveaux pour la découverte d'anomalies dans les traces de produits manufacturés. Nous présentons un ensemble de travaux sur l'extraction de motifs anormaux sous différents langages de motifs (itemsets, séquences, trajectoires, etc), puis un champ récent qui concerne la description d'anomalies. Puis, nous proposons un panorama des méthodes de découverte d'anomalies dans les données. Finalement, nous

clôturons cette section en présentant les travaux qui incluent des modèles experts pour guider la fouille de motifs. Ces modèles permettent de calculer une mesure d'intérêt pour un motif en utilisant de la connaissance experte modélisée par des fonctions mathématiques, des modèles à base de règles, ou encore des réseaux bayésiens ou des chaines de Markov. Nous montrons que l'utilisation de modèles est importante pour expliciter la connaissance experte du domaine de la traçabilité unitaire et qu'elle est pertinente pour la découverte de motifs intéressants.

Dans le second chapitre, nous montrons, avec notre première contribution, comment ces travaux peuvent être inclus dans un processus global et formalisé de valorisation des traces unitaires. Ce processus prendra la forme d'un cadre méthodologique que nous décrivons et formalisons point par point. Nous nous attelons spécifiquement à décrire l'ensemble du processus de fouille de traces unitaires, de la sélection des traces, à la conception et la mise en forme de la connaissance experte, jusqu'à l'analyse des traces, pour établir l'ensemble des degrés de liberté et des possibilités expressives du cadre méthodologique mis en place. Nous montrons la pertinence du concept de modèle de filière pour la visualisation des traces, et son utilisation pour l'extraction d'anomalies dans les traces. Ces différentes étapes sont validées par un prototype qui permet de démontrer la pertinence de ce cadre et nous présentons ses quelques principales fonctionnalités en fin de chapitre.

Le troisième chapitre nous permet d'indiquer l'importance de la modélisation de la connaissance experte sous forme de graphe statique orienté, tant au niveau de l'analyse du système manufacturier en lui même, que du support qu'il permet à la fouille d'anomalies. Nous proposons avec notre seconde contribution, une méthode permettant de classer les traces unitaires comme normales ou anormales suivant certaines des propriétés de ce graphe, avant d'extraire les ensembles de propriétés permettant de décrire les collections de traces anormales. Cette partie se termine avec la présentation d'une méthode de description des collections de traces grâce à des motifs appuyée par le modèle expert, et démontrera sa pertinence sur une base de données réelle de traces de jeu vidéo. Notre méthode peut s'appliquer à différents domaines car nous l'avons instanciée avec succès dans le domaine de l'e-sport pour l'aide aux joueurs. Nous avons publié cette contribution à la conférence Extraction et Gestion des Connaissances (EGC) 2016 à Reims [16] sous le nom *Découverte de motifs intelligibles et caractéristiques d'anomalies dans les traces unitaires*. Une publication va être présentée également dans la session spéciale Game Data Science de la conférence IEEE Data Science and Advanced Analytics (DSAA) 2016 sous le titre *What did I do Wrong in my MOBA Game ? : Mining Patterns Discriminating Deviant Behaviors* [17].

Dans un quatrième chapitre, nous proposons une troisième contribution pour un autre type de détection d'anomalies. Nous nous sommes intéressé au problème d'agents ayant plusieurs identités au sein d'un même système manufacturier, permettant de détourner des produits même lorsque ceux ci ont déjà été suspectés de fraudes. Appliquée sur des traces de jeux vidéos également, cette méthode a démontré son efficacité à la découverte de joueurs ayant plusieurs identités à partir de leurs collections de traces de jeu. Nous démontrons alors que si ces traces sont produites par des sources de prime

abord différentes, il est possible de découvrir efficacement si certaines sources sont en réalité les mêmes (les mêmes joueurs jouant sous des noms différents par exemple). Cette contribution a été présentée lors du second atelier intitulé Machine Learning and Data Mining for Sports Analytics (MLSA) lors de la conférence European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML/PKDD) 2015 à Porto (Portugal) sous le titre *Identifying Avatar Aliases in Starcraft 2* [14]. Une deuxième publication a été présentée lors de la conférence IEEE Data Science and Advanced Analytics (DSAA) 2015 sous le titre *When cyberathletes conceal their game : Clustering confusion matrices to identify avatar aliases* [15].

Chapitre 2

État de l'art

La masse des données issues de la traçabilité de produits doit être pré-traitée, agrégée et stockée dans des systèmes d'informations permettant de gérer ces grandes collections de traces unitaires efficacement. Les analystes du milieu manufacturier possèdent des outils d'interrogations classiques des données de traçabilité, sous la forme de tableaux de bord. La première partie de l'état de l'art s'attellera à décrire les problématiques spécifiques à la traçabilité de produits (captage des données, agrégation, nettoyage) et les différents types de données et protocoles connus (données RFID, GPS et EPCIS). Nous décrivons finalement les travaux s'intéressant à l'analyse des données de traçabilité, que ce soit pour l'interrogation, le stockage, la compression ou la détection d'erreurs d'enregistrement.

Une fois posé le cadre de la traçabilité et les possibilités d'interrogations existantes, nous présentons dans la suite de l'état de l'art un ensemble de méthodes de fouilles de données, utilisées dans nos contributions, en deux temps. Premièrement, nous détaillerons un ensemble de méthodes de fouille de motifs que nous avons utilisées pour notre contribution exposée dans le Chapitre 3. Plus précisément, nous avons utilisé la fouille de motifs émergents pour décrire les collections de traces jugées négatives par rapport à un modèle de filière (un modèle de référence). Nous détaillons également l'analyse de concepts formels et sa généralisation sous la forme de structures de patrons [31] qui nous permet, dans une autre contribution, de reconnaître un agent qui apparait sous plusieurs identités.

A l'issue de cet état de l'art, nous démontrons les besoins pour les industriels de la construction d'un cadre global et générique pour la fouille de traces, qui propose un processus complet depuis la réception et la transformation des traces en scénario de fouille, jusqu'à la valorisation grâce à de multiples méthodes de fouille.

2.1 Capter, stocker et interroger des données de traçabilité

Les systèmes de traçabilité produisent des volumes très importants de données. En effet, les objets transitant dans ces systèmes se comptent par millions et capter les évènements génère ainsi des milliards d'enregistrements. De plus, les traces peuvent être

constituées de nombreux évènements, par exemple dans le cas des données RFID ou GPS, un enregistrement peut être produit chaque seconde ou chaque minute. Enfin, ces données sont redondantes puisque des produits voyageant ensemble auront de nombreuses similarités.

Nous décrivons dans cette section l'objectif général de la traçabilité, ainsi que trois structures de données couramment étudiées : les données RFID, GPS et le format EPCIS. Nous concluons par les différents problèmes que soulèvent la traçabilité : la collecte et le stockage des données, les capacités d'interrogation des données et la recherche d'anomalies. Nous pouvons alors pointer l'absence de méthodes généralistes adaptées à la description de phénomènes et d'anomalies.

2.1.1 La problématique de la traçabilité

De la création des produits en passant par la distribution jusqu'à la vente, de nombreux acteurs différents interagissent. De même, la bonne circulation des produits est fortement dépendante de divers facteurs matériels, géographiques, humains ou météorologiques : état des véhicules et des machines, bon fonctionnement des moyens d'acheminements par route, avion ou bateau, absence d'accidents, personnel formé et en nombre suffisant, conditions climatiques favorables (risques de verglas, tempêtes, orages, ... pouvant freiner ou empêcher la distribution).

Pour informer les experts de l'acheminement de leurs produits et avoir des informations précises sur les processus métiers, des systèmes de traçabilité ont été mis en place. De manière générique, la traçabilité de produits concerne un unique produit qui possède un identifiant et dont la séquence d'évènements correspond à la suite d'actions que le système va capter lors de son acheminement le long du processus global. Ces évènements possèdent des informations variées, dépendantes du système de captage mis en place et du protocole utilisé. La traçabilité concerne plusieurs aspects de la gestion des données comme l'aspect physique avec l'ensemble des capteurs, des réseaux de communication et des serveurs de stockage des données ; l'aspect logiciel de gestion des données avec l'ensemble des protocoles et programmes de captage, transmission, gestion, stockage et restitution des données de traçabilité ; et enfin l'aspect analyse de données avec l'ensemble des programmes permettant l'analyse des données de traçabilité stockées.

L'objectif final est de valoriser ces traces pour extraire des informations permettant de détecter des évènements intéressants pour les experts ou encore d'appliquer des algorithmes d'interrogation de type SQL ou OLAP pour produire des analyses par agrégation de données.

Nous présentons par la suite trois cas de données de traçabilité (RFID, GPS, EPCIS) avec pour chaque cas des exemples précis de travaux exploitant ces données.

2.1.2 Données de traçabilité

Les données de traçabilité possèdent des caractéristiques et attributs fortement dépendants des applications, des types de capteurs et de la capacité de gestion du volume des données. En effet, les capteurs peuvent uniquement donner la position d'un objet

au cours du temps si l'on analyse les trajectoires spatiales de ceux-ci, ou alors rester fixes mais donner des informations selon plusieurs dimensions (température, pression atmosphérique), ou encore combiner les deux aspects. Nous présentons trois types de données : les données RFID captées via des tags RFID très utilisés dans l'industrie manufacturière ; les données GPS servant en grande partie à la traçabilité de véhicules (taxi, voiture, avion, bateau) ou d'animaux (analyse de mouvements d'oiseaux) ; les données EPCIS, moins répandues mais émergentes dans le domaine manufacturier et au cœur du contexte industriel dans lequel a été réalisé ce travail.

2.1.2.1 Données RFID

Les données RFID sont fournies par des tags RFID (voir Photo 2.1) pouvant être collés ou intégrés dans des objets ou des corps organiques. Ces étiquettes, appelés transpondeurs, possèdent une antenne associée à une puce électronique qui permet de recevoir et de répondre à des requêtes envoyées via un émetteur-récepteur. Ces tags identifient chaque objet et sont lisibles à distance, permettant alors de tracer un grand nombre de produits. Aggrawal et Han [2] proposent un état de l'art sur l'analyse des données RFID. Ils discutent de l'ensemble des domaines utilisant ce type de système : la traçabilité de produits en temps réel, l'Internet des Objets, les applications médicales, les systèmes de paiement, les contrôles d'accès, les mouvements d'animaux, les prêts de livres, la gestion des bagages dans les aéroports ou encore le démarrage des voitures. Les problématiques principales sont multiples. Premièrement les volumes de données sont très importants puisque les lecteurs vont enregistrer des évènements à intervalles réguliers potentiellement courts. De plus, les données peuvent être bruitées et redondantes. En effet, les tags peuvent être détériorés et lus par plusieurs lecteurs à la fois à de multiples instants, générant encore plus de données. La masse de données rend le processus de nettoyage encore plus difficile. La sécurité et la vie privée sont également un enjeu important, étant donné que les tags peuvent par exemple être implantés dans différents objets du quotidien, et tracer des personnes sans leur consentement. Finalement, la technologie RFID, ainsi que les lecteurs, sont vulnérables aux pannes physiques et logistiques, ce qui rend la collecte d'informations difficile car des évènements peuvent manquer, ce qui peut entrainer des résultats erronés pour certains systèmes d'analyse.

Figure 2.1 – Photo d'un tag RFID

2.1.2.2 Données GPS

Le système GPS (Global Positioning System) est un système de géolocalisation mondial qui s'appuie sur les signaux radios émis par des satellites spécialement dédiés. Auparavant destiné au domaine militaire, les applications du GPS se sont répandues rapidement, puisqu'il est possible maintenant de connaître la position d'objets ou d'individus de manière précise sous condition que le récepteur soit équipé du matériel adéquat. Les données GPS sont caractéristiques des trajectoires spatio-temporelles, issus de déplacements de personnes, d'animaux, de véhicules ou de l'observation de phénomènes météorologiques. Une trajectoire spatiale est une trace générée par un objet se déplaçant dans un espace géographique, et représentée par une séquence ordonnée de points comme $s = p_1 \rightarrow p_2 \rightarrow ... \rightarrow p_n$ où chaque point possède des coordonnées spatio-temporelles comme $p_i = (x_i, y_i, t_i)$. Zheng [98] propose un état de l'art sur la fouille de trajectoires spatio-temporelles. Ce champ comporte de nombreuses tâches. Premièrement il faut prétraiter des trajectoires (détection d'arrêts, filtrage du bruit, segmentation, compression). Il s'agit, en amont, de gérer le bruit présent dans les trajectoires, à cause des contraintes inhérentes aux données GPS. En effet, le signal peut être mauvais et ainsi mal (voir pas du tout) capter la position du récepteur. De plus, lorsque des millions d'objets sont tracés, les données GPS deviennent très volumineuses, suivant la durée de l'intervalle auquel on enregistre les données de positions. Des techniques de réduction de la taille des trajectoires (en réduisant le nombre de points par exemple) peuvent permettre de réduire la taille totale de la base. Il est aussi utile de détecter si une séquence consécutive de points est caractéristique du stationnement d'un objet, pour offrir une sémantique différente des autres points de la trajectoire. La grande masse de données des trajectoires a ouvert la voie vers la proposition de méthodes spécifiques de fouille de trajectoires, permettant de regrouper les trajectoires selon leurs similitudes, d'y découvrir des motifs périodiques, des sous-trajectoires fréquentes, des points d'intérêt (des zones où passent et s'arrêtent souvent les objets), ou encore la découverte de trajectoires anormales.

2.1.2.3 Données EPCIS

EPCIS [4] est un standard axé sur la mise en place d'applications pour créer et partager des données sous la forme d'enregistrements en dehors ou au sein d'entreprises. L'objectif final du partage global de ces données est de permettre aux utilisateurs d'avoir des informations sur l'objet ainsi qu'une vue globale de son cycle de vie au sein d'un contexte industriel particulier.

Le Diagramme 2.2 représente les objets importants du standard EPCIS ainsi que leurs relations. Un évènement EPCIS (table EPCISEvent) est un tuple composé d'une estampille temporelle (eventTime) et peut posséder plusieurs types (ObjectEvent, AggregationEvent,TransactionEvent, TransformationEvent). Dans notre cas d'analyse, les analystes mettant en place la base de données EPCIS utilisent uniquement les évènements ObjectEvent et AggregationEvent. Dans le premier cas, un évènement est assigné à un identifiant unique de produits appelé EPC (Electronic Product Code). Le champ

4. `http://www.gs1.org/docs/epc/epcis_1_1-standard-20140520.pdf`

action désigne un type d'action (CREATION,TRANSFORMATION,DESTRUCTION), le champ 'bizstep' correspond à une étape dans le système manufacturier (PACKING, FILLING, COMMISSIONING, DECORATING, SHIPPING, etc...), les champs 'readpoint' et 'bizLocation' déterminent les localisations de l'enregistrement de l'évènement (point de lecture, et bâtiment). L'évènement AggregationEvent permet de modéliser un évènement d'agrégation, c'est-à-dire lorsque plusieurs objets sont conditionnés dans une palette ou un carton en vue d'une expédition, ou quand plusieurs ingrédients sont mélangés, dans le cas de l'agro-alimentaire. Il existe plusieurs niveaux d'agrégations possibles, par exemple des produits sont conditionnés dans un carton eux mêmes réunis en palette. La Table 2.1 présente l'ensemble des traces pour un produit. Elle est constituée des évènements assignés au produit ayant l'identifiant 1 mais aussi aux palettes qui vont le contenir (Objets 4 et 5). PACKING et UNPACKING sont deux évènements d'agrégations et les autres évènements sont des ObjectEvent. Les contraintes industrielles génèrent des comportements très caractéristiques, avec le regroupement de produits à une localisation puis une dispersion par la suite. Dans le cas général de la distribution de produits, cette dispersion est limitée géographiquement puisque les distributeurs s'engagent par contrat à la distribution de produits dans des zones limitées. Ainsi, des produits qui sont souvent ensembles sont susceptibles d'être vendus dans la même zone géographique.

ev_id	epc_id	id_lot	epc_parent	time	biz_step	biz_loc
1	1	X	-1	2016-01-01 05 :11 :33	COMMISSIONNING	loc_A
4	4	X	-1	2016-01-01 05 :11 :33	COMMISSIONNING	loc_A
5	1	X	4	2016-01-01 05 :25 :20	PACKING	loc_A
9	4	X	-1	2016-01-01 05 :36 :45	STORING	loc_A
10	4	X	-1	2016-01-02 13 :05 :10	SHIPPING	loc_A
11	4	X	-1	2016-01-04 16 :25 :46	RECEIVING	loc_B
12	4	X	-1	2016-01-04 16 :25 :46	UNPACKING	loc_B
13	4	X	-1	2016-01-04 16 :25 :46	DECOMMISSIONNING	loc_B
14	1	X	4	2016-01-04 16 :37 :20	FILLING	loc_B
17	5	X	-1	2016-01-04 16 :47 :15	COMMISSIONNING	loc_B
18	1	X	5	2016-01-04 16 :47 :15	PACKING	loc_B
21	5	X	-1	2016-01-04 16 :47 :15	PACKING	loc_B
22	5	X	-1	2016-01-05 06 :12 :33	SHIPPING	loc_B
23	5	X	-1	2016-01-08 13 :22 :01	RECEIVING	loc_C
24	5	X	-1	2016-01-08 13 :22 :01	UNPACKING	loc_C
25	5	X	-1	2016-01-08 13 :22 :01	DECOMMISSIONNING	loc_C
26	1	X	-1	2016-02-12 11 :01 :25	RETAIL_SELLING	loc_C

TABLE 2.1 – Un exemple d'une trace unitaire

2.1.3 Problèmes et solutions

Nous décrivons les problématiques globales de la collection et du stockage des données de traçabilité, de l'interrogation de ces données, ainsi que l'extraction d'anomalies et d'erreurs d'enregistrements.

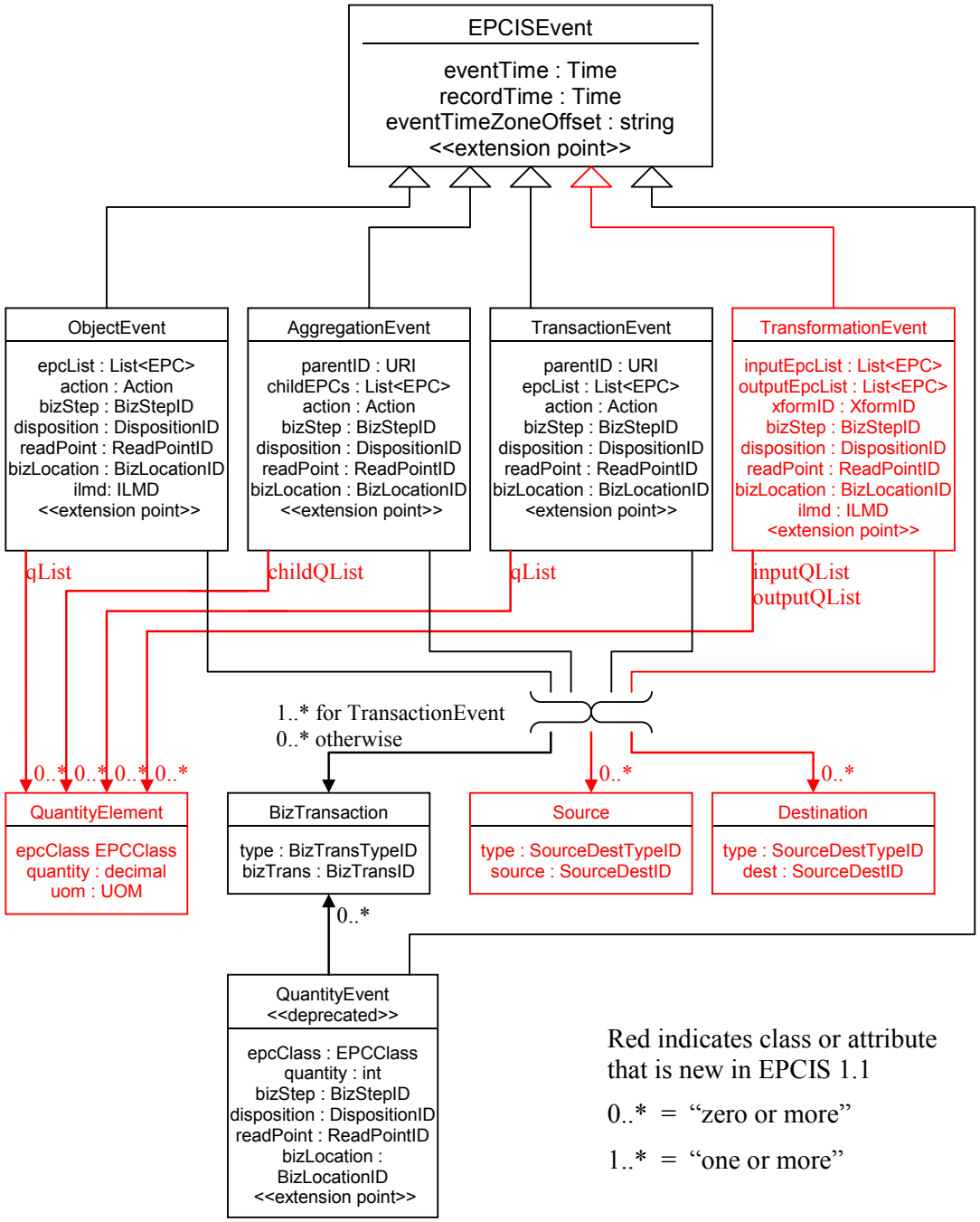


FIGURE 2.2 – Description des notions utilisées par le protocole EPCIS

2.1.3.1 Collecte, stockage et interrogation des données

Pour répondre aux caractéristiques spécifiques des données RFID et permettre leur exploitation par des méthodes d'analyse, des travaux ont été proposés pour améliorer la

qualité ou réduire le volume des données. Gonzalez et Han [35] [34] proposent un point de vue général sur les modèles permettant de stocker et compresser les données RFID puis de les analyser grâce à un système de requêtes avancé. L'analyse de ces données constitue un réel défi puisque le volume de données est très important à cause de la redondance et du bas niveau d'abstraction. Les article proposent de réduire la redondance des données RFID en les regroupant. Par exemple, si le lecteur de tags RFID enregistre des tuples de la forme $(EPC, location, time)$ à chaque intervalle de temps fixe quand l'objet est immobile, alors de nombreux tuples redondants sont générés. Les auteurs proposent de les regrouper sous la forme d'un tuple $(EPC, location, time\_in, time\_out)$ où $time\_in$ est le moment où l'objet entre dans le lieu et $time\_out$ le moment où il en sort. Une autre proposition est faite pour regrouper les objets qui bougent et restent ensemble tout au long du système manufacturier. Une architecture nommé RFID-CUBOID [35] basée sur les cubes de données [28] est présentée. Elle étend les cubes de données en permettant l'analyse d'objets mobiles selon plusieurs dimensions. Il est possible par exemple de connaître le temps moyen de stockage pour un type spécifique de produits ayant parcourus des sites précis. Une table permet de regrouper les données des objets restant ensemble et une autre table permet de relier les objets voyageant ensemble. Il est alors possible d'agréger les données d'objets suivant leurs comportements, par exemple pour répondre à la question : "Quel est le temps moyen de stockage des produits de type A sur le site industriel X durant l'année 2016 ?" ou encore "Quel est le temps moyen mis par le distributeur Z pour recevoir ses produits, les stocker puis les envoyer à ses vendeurs durant l'année 2015 ? ".

2.1.3.2 Anomalies et erreurs d'enregistrement

La technologie RFID est de plus en plus répandue dans le domaine du système manufacturier. Elle permet de suivre précisément les objets et des méthodes pour détecter des anomalies sur la chaîne de production ont vu le jour. Pour détecter des anomalies comme des actions frauduleuses ou un système de vente non efficace, Masciari [68] propose un modèle d'extraction d'anomalies dans les données RFID. Il constate que les produits manufacturés sont regroupés ensemble en grande quantité dans les centres de distribution et en petite quantité dans les magasins et exploite cette caractéristique pour construire sa technique. Il propose de détecter des anomalies dans une séquence d'évènements en calculant sa similarité par rapport à une séquence d'évènements attendue (qui doit alors être construite manuellement). La mesure de similarité est la distance entre les transformées de Fourier discrètes des deux séquences d'évènements. Si la mesure dépasse un seuil, alors une alerte est levée. Cependant, cette mesure est adaptée pour comparer les séquences deux à deux uniquement, elle n'est donc pas appropriée pour traiter de grands volumes de traces.

2.2 Fouille de motifs

Dans cette section, nous exposons les différentes techniques pouvant décrire des collections de traces unitaires. En effet, une fois récupérées, stockées et pré-traitées, nous cherchons à valoriser ces collections de traces pour extraire de la connaissance intéressante. Nous nous intéressons dans un premier temps, à décrire ces collections de traces.

Nous subdivisons les techniques de description en trois grands groupes : les techniques permettant la découverte d'évènements intéressants dans ces traces ; les travaux permettant de valoriser les traces via des solutions ad-hoc, c'est-à-dire s'attachant à extraire un type d'évènements très précis ; puis enfin une ouverture vers les processus de fouille de données. Nous nous focaliserons sur cette dernière partie, en détaillant les différents solveurs qui peuvent être appliqués aux données de traçabilité et le type de connaissance que l'on peut en extraire. Pour finir cette section, nous nous demandons comment exprimer la connaissance experte que l'on peut avoir en amont de la traçabilité de données. Concrètement, il s'agira d'étudier les travaux utiles à la modélisation de cette connaissance, via des modèles construits manuellement par les experts ou induits par un ensemble de données.

Les premiers algorithmes de fouille de données s'attachent à découvrir des motifs dans des bases de données transactionnelles (ensemble de propriétés non ordonnées). Nous rappelons l'exemple que nous avions pris dans l'introduction avec la Table 2.2.

	s_1	s_2	s_3	s_4
p_1	×	×		
p_2	×	×	×	
p_3	×		×	×
p_4	×		×	×

Table 2.2 – Exemple de données

Chaque ligne de la table est une instance, un objet ou une transaction, décrit par des propriétés booléennes, les colonnes. Si une croix est dans une cellule, alors l'objet de la ligne vérifie la propriété de la colonne. Nous pouvons aussi adopter une notation plus succincte, mais visuellement moins pratique, où l'on ne conserve que les propriétés vraies pour chaque objet, par exemple p_2 sera rattachée à l'ensemble $\{s_1, s_2, s_3\}$. Supposons que l'on cherche des ensembles qui apparaissent au moins deux fois dans les données (le nombre d'apparitions étant appelé le support), alors on observe que l'on a les ensembles fréquents suivants : $\{s_1\}, \{s_2\}, \{s_3\}, \{s_4\}, \{s_1, s_2\}, \{s_1, s_3\}, \{s_3, s_4\}, \{s_1, s_3, s_4\}$. L'ensemble $\{s_1, s_2, s_3\}$ n'est pas fréquent car il n'est présent que dans la description de p_2, son support est de 1. Le volume de données en sortie de l'analyse étant important, la définition de préférences utilisateurs est importante pour cibler la pertinence des résultats. En effet, le nombre de motifs extraits est généralement supérieur à la taille de la base de données, par exemple dans des bases de données transactionnelles, le nombre de motifs possibles est de $2^n - 1$ avec n le nombre d'items dans la base ce qui est généralement largement supérieur au nombre de transactions dans la base. Ainsi, la définition de contraintes

sur les motifs devient nécessaire, par exemple pour ne conserver que les ensembles qui apparaissent souvent, ou très peu dans les données.

La contrainte de fréquence exposée dans Agrawal et Skirant [3] est souvent exploitée durant les processus de fouille de données, ce qui permet d'économiser en temps de calcul et en mémoire (l'algorithme retourne uniquement les motifs qui apparaissent au moins un certain nombre de fois dans les données). La fouille sous contraintes (Boulicaut et Jeudy [12]) permet de spécifier la pertinence d'un motif par la définition de propriétés que ce motif doit vérifier. Par exemple, la contrainte de fréquence indique que les motifs doivent apparaître un certain nombre de fois dans les données pour être considéré comme pertinent. Les propriétés mathématiques de ces contraintes peuvent également permettre d'optimiser les calculs en n'explorant pas entièrement toutes les combinaisons. Il s'agit alors de répondre conjointement à deux problématiques : la qualité des motifs d'une part ; et assurer une extraction complète dans des temps raisonnables (passage à l'échelle) sur des données souvent nombreuses et parfois complexes (graphes attribués et dynamiques par exemple) d'autre part.

Un cadre formel pour la fouille de données sous contraintes a été proposé par Mannila et al. [65]. Soit une base de données $\mathcal{D}$ qui possèdent une description exprimée par un langage de motifs $\mathcal{L}$ (itemsets, séquences, trajectoires,...), et une conjonction de contraintes $\mathcal{C}$ qui permettent de dire si un motif $l \in \mathcal{L}$ de $\mathcal{D}$ est pertinent, la fouille sous contrainte consiste à calculer la théorie :

$$Th\left(\mathcal{D},\mathcal{L},\mathcal{C}\right) = \{\phi \in \mathcal{L} | \mathcal{C}\left(\phi,\mathcal{D}\right) = true\} \tag{2.1}$$

De nombreux travaux ont proposés, avec différents langages de motifs $\mathcal{L}$ ayant des structures permettant de capturer différents aspects des données (co-occurences, ordre, temporalité, spatial,...), des contraintes pour déterminer la pertinence des motifs en définissant des propriétés attendues.

Les sections suivantes vont présenter une sélection de différents langages de motifs et, pour chacun, présenter les types de contraintes communément étudiés. Nous décrivons pour chaque langage les aspects des données qu'il décrit, avec un exemple sur des données de traçabilité transformées.

2.2.1 Fouille de motifs ensemblistes

Les données transactionnelles correspondent à une table binaire, où les transactions représentent les objets et les colonnes des propriétés booléennes. Si une cellule de la table possède une croix (ou une valeur booléenne égale à "vrai") alors cela veut dire que l'objet correspondant à la transaction vérifie la propriété de la colonne correspondante. Dans notre cas, les transactions représentent les produits manufacturés et les propriétés sont générées à partir des traces ECPIS (ce point sera particulièrement abordé dans le chapitre suivant), suivant la connaissance que l'on veut modéliser et analyser. Par exemple, à partir de la Table 1.4 présentée dans l'introduction, nous encodons des propriétés pour produire la table binaire présentée dans la figure 2.3. Les propriétés *loc_A*, ..., *loc_E* sont vraies si le produit est passé par la localisation correspondante. On génère

aussi deux propriétés spécifiques au domaine de la traçabilité manufacturière : *vente* qui est vraie si le produit a été vendu (présence de l'évènement $RETAIL\_SELLING$), $stockage\_usine\_ > \_48h$ si le produit a été stocké à un moment de son cycle de vie plus de 48h en usine.

epc_id	*loc_A*	*loc_B*	*loc_C*	*loc_D*	*loc_E*	*vente*	$stockage\_usine\_ > \_48h$
1	×	×	×			×	
2	×	×	×				
3	×	×	×			×	
6	×			×	×	×	×
7	×			×	×		×
8	×			×	×	×	×

FIGURE 2.3 – Collection de traces unitaires sous la forme d'une table binaire

De nombreux solveurs, comme CHARM [96] ou LCM [89], ont été proposés pour découvrir des motifs dans ces tables binaires : co-occurrences de propriétés fréquentes, aussi appelés *itemsets* (avec des représentations condensées pour réduire le nombre de résultats), rares, ou ayant des contraintes spécifiques (par exemple le nombre de propriétés maximales). Des travaux se sont intéressés également à la découverte de motifs avec des transactions auxquelles sont assignées deux classes (positives ou négatives) comme les motifs émergents [25]. L'objectif est de découvrir les motifs qui sont caractéristiques d'une des deux bases (ou plus) selon une mesure spécifique (mesures statistiques, basées sur le support, etc.). Nous présentons par la suite, une sélection de ces travaux, ainsi qu'une définition des différentes contraintes et mesures.

Nous établissons deux définitions de base communes à l'ensemble des travaux présentés :

Definition 1 (Itemset) *Un itemset I est un ensemble non vide de k items qui est un sous-ensemble de $\mathcal{I}$ avec $\mathcal{I} = \{i_1, i_2, ..., i_m\}$ l'ensemble fini de tous les items.*

Definition 2 (Base de transaction) *Une base de transactions $\mathcal{D} = \{t_1, t_2, ..., t_n\}$ est un ensemble de n transactions, chacune identifiée par un unique TID. Chaque transaction t consiste en un ensemble d'items I de $\mathcal{I}$.*

Exemple 1 *Dans la Figure 2.3, nous avons 6 transactions :*
$t_1 = \{loc\_A, loc\_B, loc\_C, vente\}$,
$t_2 = \{loc\_A, loc\_B, loc\_C\}$,
$t_3 = \{loc\_A, loc\_B, loc\_C, vente\}$,
$t_4 = \{loc\_A, loc\_D, loc\_E, vente, stockage\_usine\_ > \_48h\}$,
$t_5 = \{loc\_A, loc\_D, loc\_E, stockage\_usine\_ > \_48h\}$ *et*
$t_6 = \{loc\_A, loc\_D, loc\_E, vente, stockage\_usine\_ > \_48h\}$.

Exemple 2 *Dans la Figure 2.3, les 6 itemsets correspondent à un ensemble nommé $\mathcal{D}$ appelée base de transaction soit $\mathcal{D} = \{t_1, t_2, t_3, t_4, t_5, t_6\}$.*

2.2.1.1 Motifs fréquents

L'une des premières problématiques de l'exploration combinatoire des propriétés booléennes portait sur l'extraction d'ensemble d'items fréquemment observés dans une base de transaction. La fouille d'itemsets fréquents a été proposée avec l'algorithme Apriori [3] pour extraire les ensembles de produits fréquemment achetés ensemble dans un magasin. Il s'agit d'extraire tous les ensembles de propriétés fréquents observés dans au moins un certain nombre de transactions. Le nombre d'ensembles possibles est de $2^n - 1$ où n est le nombre d'items, ce qui en fait un problème difficile à résoudre. Les définitions suivantes permettent de décrire la problématique de la fouille de motifs fréquents.

Definition 3 (Support absolu d'un itemset) *Le support absolu d'un itemset X, noté $sup(X)$ est le nombre de transactions dans $\mathcal{D}$ qui contiennent X soit :*

$$supp(X) = |\{X' | X' \in \mathcal{D}, X \subseteq X'\}| \tag{2.2}$$

Definition 4 (Fréquence d'un itemset) *La fréquence d'un itemset I est le rapport du nombre de transactions supportant X sur le nombre total de transactions de la base $\mathcal{D}$ soit :*

$$freq(X) = \frac{supp(X)}{|\mathcal{D}|} \tag{2.3}$$

Definition 5 (Itemset fréquent) *Un itemset X est fréquent si son support est supérieur à un seuil de fréquence minimum min_freq prédéfini.*

Problème 1 *Soit une base de transaction $\mathcal{D}$ et un seuil de fréquence minimum min_freq, trouver l'ensemble correct et complet des itemsets fréquents est appelé le problème d'extraction d'itemsets fréquents.*

Exemple 3 *Soit $\mathcal{D}_1$ la base de transactions présentée dans le Tableau 2.3 on $X_1 = \{loc\_A\}$,$X_2 = \{loc\_A, loc\_B\}$ avec $freq(X_1) = 1$ et $freq(X_2) = 0.5$. Avec un seuil de fréquence de 0.6, l'ensemble complet des motifs est $\{loc\_A\}$ et $\{vente\}$.*

L'idée principale d'Apriori est de générer itérativement l'ensemble des itemsets candidats de longueur $k+1$ à partir de l'ensemble des itemsets fréquents de longueur k et de vérifier que les fréquences des occurrences dans la base de transactions soient supérieures à *min_sup*. Il est basé sur la propriété d'anti-monotonie du support : si un itemset de longueur k est non fréquent, les itemsets de longueur $k+1$ qui l'incluent ne peuvent pas être fréquents. Si les algorithmes basés sur Apriori possèdent de bonnes performances en réduisant la taille des ensembles candidats, ils ne sont pas efficaces quand il y a de nombreux itemsets fréquents, des itemsets fréquents longs ou un seuil de fréquence minimum bas. Par exemple s'il y a 10^4 itemsets fréquents de taille 1, alors 10^8 candidats de taille 2 sont générés. De plus, ces algorithmes parcourent de nombreuses fois les mêmes candidats. FP-growth [40] propose de réduire le temps d'exécution en diminuant la taille

de la base de transaction en stockant une version compressée sous forme d'arbre (FP-tree). Des algorithmes très efficaces utilisent des structures de données particulières pour extraire rapidement des versions condensées des itemsets fréquents (que l'on décrit dans la section suivante), comme CHARM [96] et LCM [89].

Lorsque les itemsets sont extraits, il est possible de calculer des règles d'associations [3][40] [95] à partir des itemsets qui permettent de décrire des rapports de co-occurrences et de dépendance entre les itemsets.

Definition 6 (Règle d'association) *Une règle d'association est une expression* $X \rightarrow Y$ *où* X *et* Y *sont des itemsets,* $X, Y \subset \mathcal{D}$ *et* $X \cap Y = \emptyset$.

Definition 7 (Support d'une règle d'association) *Le support absolu supp d'une règle d'association est le nombre de transactions où* X *et* Y *apparaissent ensemble, soit :*

$$supp(X \rightarrow Y) = sup(X \cup Y) \tag{2.4}$$

Definition 8 (Fréquence d'une règle d'association) *Le support relatif est le rapport entre le nombre de transactions où* X *et* Y *apparaissent ensemble sur le nombre de transactions total, soit :*

$$\frac{sup(X \cup Y)}{|\mathcal{D}|} = P(X \wedge Y) \tag{2.5}$$

Exemple 4 *Une règle d'association* $\{loc\_A, loc\_D, loc\_E\} \rightarrow \{stockage\_usine\_ > \_48h\}$ *avec un support de 3 et une fréquence de 50% désigne que la proportion des transactions qui contiennent à la fois* $\{loc\_A, loc\_D, loc\_E\}$ *et* $\{stockage\_usine\_ > \_48h\}$ *est de 50%.*

Cette notion permet d'extraire des relations de co-occurences entre les itemsets, en indiquant les évènements qui sont susceptibles de survenir ensemble. Une notion importante, la confiance, permet de savoir à partir d'un itemset donné la probabilité d'observer qu'une transaction donnée possède un autre itemset.

Definition 9 (Confiance d'une règle d'association) *La confiance d'une règle d'association est la probabilité conditionnelle que, si une transaction contient* X *alors elle contient* Y*, soit :*

$$conf(X \rightarrow Y) = P(Y|X) = \frac{sup(X \cup Y)}{sup(X)} \tag{2.6}$$

Exemple 5 *Comme* $supp(\{loc\_A, loc\_D, loc\_E\}) = 3$ *et* $supp(\{stockage\_usine\_ > \_48h\}) = 3$ *alors la confiance de notre règle précédente est de 100%. Par exemple, la confiance de la règle* $\{loc\_A, loc\_B, loc\_C\} \rightarrow \{vente\}$ *est de 66%, c'est-à-dire que les deux tiers des produits qui sont passés par* loc_A, loc_B, loc_C *ont été vendus.*

Avec ces définitions, on peut finalement formaliser deux types de règles : les règles fréquentes, dont la fréquence est supérieure à un seuil min_sup, et les règles fortes, où la confiance, en plus, est supérieure à un seuil min_conf.

Definition 10 (Règle fréquente) *Une règle est fréquente si $X \cup Y$ est un motif fréquent, c'est-à-dire si $sup(X \cup Y) \geq min\_sup$.*

Definition 11 (Règle forte) *Une règle $X \rightarrow Y$ est forte si $conf(X \rightarrow Y) \geq min\_conf$ où min_conf est un seuil de confiance minimale spécifié par l'utilisateur.*

Pour pouvoir procéder à l'extraction de règles d'associations fréquentes avec une confiance haute, il faut tout d'abord extraire les itemsets fréquents dans $\mathcal{D}$ par rapport à un seuil min_freq. Puis, à partir de l'ensemble des itemsets fréquents et du seuil de confiance minimale min_conf, le problème d'extraction de règles d'association est de trouver toutes les règles fréquentes et fortes. L'inconvénient est que de nombreux ensembles et règles fréquents vont être retournés. Par exemple, $\{loc\_A, loc\_B\} \rightarrow \{vente\}$ et $\{loc\_A, loc\_B, loc\_C\} \rightarrow \{vente\}$ seront retournées alors que la première règle est contenue dans l'autre.

Pour réduire le nombre de motifs redondants, des représentations condensées sans perte d'information ont été proposées.

2.2.1.2 Motifs fermés et maximaux

Les représentations condensées d'ensembles permettent de ne récupérer que les itemsets les plus spécifiques c'est-à-dire ceux dont le maximum d'items est partagé par un maximum de transactions. Certains motifs sont inclus dans d'autres tout en ayant la même mesure, ce qui provoque une redondance d'information. Ainsi les itemsets fermés ont été introduits dans Pasquier et al.[72] pour supprimer ces motifs redondants. Notons que la définition des concepts formels, issus de l'analyse de concepts formels et présentée par la suite, correspondent à des itemsets fermés.

Definition 12 (Itemsets fermés) *Un itemset X est un itemset fermé s'il n'existe pas $X' \subseteq X$ et tel que $supp(X') = supp(X)$.*

Les itemsets maximaux correspondent aux itemsets clos fréquents les plus spécifiques et permettent de réduire encore plus le nombre de motifs.

Definition 13 (Itemsets maximaux) *Soit $\mathcal{F}$ l'ensemble des itemsets fréquents, alors un itemset $X \in \mathcal{F}$ est un itemset maximal fréquent s'il vérifie les contraintes suivantes : $\nexists X' \in \mathcal{F}$ tel que $X \subset X'$.*

Exemple 6 *Dans l'exemple 2.3, nous avons ces motifs fermés :*
$\{loc\_A\}, \{vente\}, \{loc\_A, loc\_B, loc\_C\}, \{loc\_A, loc\_B, loc\_C, vente\}$,
$\{loc\_A, loc\_D, loc\_E, stockage\_usine\_ > \_48h\}$ *et*
$\{loc\_A, loc\_D, loc\_E, vente, stockage\_usine\_ > \_48h\}$.

De nombreux travaux ont été proposés pour extraire de tels motifs. L'algorithme DCI_Closed [63] s'attaque au problème de duplication des itemsets clos, et retire rapidement les itemsets dupliqués sans garder en mémoire l'ensemble des itemsets. CloStream

[94] permet d'extraire des itemsets clos dans un flux de données où des transactions sont ajoutées ou retirées au fil du temps. CHARM [96] propose plusieurs améliorations dans la recherche d'itemsets clos. Il explore simultanément l'espace des itemsets et des transactions, contrairement aux méthodes précédentes qui exploitent seulement l'espace des itemsets. LCM [89], dans sa version 3, exploite les structures de données utilisées dans les précédentes études sur la fouille de motifs : les bitmaps, les arbres de préfixes et les listes. Suivant la densité de la base, chaque structure a ses avantages et ses inconvénients. LCM propose alors de combiner les trois structures pour permettre une performance optimisée dans n'importe quels cas.

La découverte de motifs fermés et maximaux a permis de réduire fortement le nombre de motifs extraits. Cependant, dans certains cas d'application, la spécification de la contrainte de fréquence minimale n'est pas simple ou pertinente. Des mesures d'intérêt pour les itemsets ont été proposées pour utiliser de la connaissance experte durant le processus même de fouille, comme Flouvat et al. [29], lors de l'exploration des différentes possibilités d'énumération. Le modèle peut servir comme référence de ce qui est attendu, ou comme fonction ayant pour variable des attributs de l'itemset et comme résultat une valeur d'intérêt.

2.2.1.3 Motifs avec des contraintes de modèles

Nous avons vu précédemment des contraintes de fréquence ou de clôture pour déterminer si un motif est intéressant ou non. Cependant, ces contraintes peuvent être difficiles à fixer et n'utilisent pas de connaissances expertes. Plusieurs travaux ont proposé de se servir de modèles comme base pour définir des contraintes d'élagage de motifs.

Flouvat et al. [29] présente un contexte spécifique composé d'experts en érosion des sols. Ils possèdent des modèles mathématiques pour estimer le risque d'érosion en fonction d'attributs numériques ou catégoriels (végétation, type de sol, pente, ...). Ces attributs sont utilisés comme variables d'entrée du modèle mathématique sous forme de fonction pour déterminer la pertinence d'un motif. Il a été démontré que les motifs sont beaucoup moins nombreux en plus d'être davantage pertinents.

Jaroszewicz et al. [46] formalisent la connaissance experte à l'aide d'un réseau bayésien qui modélise les relations causales entre les attributs et qui est construit en amont à partir de données. Ils utilisent ce modèle dans une mesure pour découvrir si un itemset est intéressant ou pas en calculant la divergence entre la fréquence du motif dans les données par rapport à la fréquence attendue calculée à partir du réseau bayésien.

Kaytoue et al. [49] proposent une méthode de fusion d'information avec une application en agronomie, où une formule mathématique permet d'évaluer la qualité d'un sol. Le problème est que ces experts donnent des paramètres différents, et les auteurs cherchent les motifs qui donnent les meilleurs combinaisons de paramètres au regard de la formule.

epc_id	*loc_A*	*loc_B*	*loc_C*	*loc_D*	*loc_E*	*vente*	*stockage_usine_ > _48h*	classe
1	×	×	×			×		+
2	×	×	×					+
3	×	×	×			×		+
6	×			×	×	×	×	-
7	×			×	×		×	-
8	×			×	×	×	×	-

TABLE 2.3 – Collection de traces unitaires labellisées, sous la forme d'une table binaire

2.2.1.4 Motifs émergents, corrélés et discriminants

Nous abordons dans cette section le cas où, à chaque transaction, on assigne une classe parmi deux (positive ou négative), ce qui permet de mettre à jour notre exemple comme présenté dans la Table 2.3. La détermination de la classe (négative ou positive) peut être manuelle ou selon des critères ou contraintes qui restent à définir. Ces motifs permettent de capturer les différences entre deux classes en étudiant les différences de support d'un motif d'une base de transactions à une autre. Les motifs émergents ont été introduits par Guozhu Dong et Jinyan Li [25]. Pour définir les motifs émergents, le taux de croissance d'un itemset I doit être calculé. Novak et al. [71] a montré que ces motifs se trouvent sous divers noms (motifs contrastés, découverte de sous-groupes) dans de multiples domaines de la fouille de données.

Definition 14 (Taux de croissance) *Soient $\mathcal{D}_1$ et $\mathcal{D}_2$ deux bases de transactions et X un itemset, alors le taux de croissance de X de $\mathcal{D}_1$ vers $\mathcal{D}_2$ $GrowthRate(X)$ est défini par :*

$$GrowthRate(X) = \begin{cases} 0, & si\ supp_{\mathcal{D}_1}(X) = 0\ et\ supp_{\mathcal{D}_2}(X) = 0 \\ \infty & si\ supp_{\mathcal{D}_1}(X) = 0\ et\ supp_{\mathcal{D}_2}(X) \neq 0 \\ \frac{supp_{\mathcal{D}_2}(X)}{supp_{\mathcal{D}_1}(X)} & sinon \end{cases}$$

Definition 15 (Motif émergent) *Soit θ un seuil d'émergence, un itemset X est un motif émergent si $GrownthRate(X) \geq \theta$.*

Exemple 7 *Nous cherchons un motif émergent caractéristique de la base négative ($\mathcal{D}^-$). Nous voyons que $GrownthRate(\{loc\_A, loc\_D, loc\_E, stockage\_usine\_ > \_48h\}) = \infty$ c'est-à-dire que l'ensemble des traces négatives passent par les localisations loc_A,loc_D, loc_E et les produits correspondants ont été stockés plus de 48h en usine.*

L'exemple précédent montre un cas particulier de motif émergent présenté dans Li et al. [59], les Jumping Emerging Pattern (JEP). La notion de JEP a été proposée car ils permettent de représenter des contrastes extrêmes entre deux bases de transactions puisqu'ils vont être présents uniquement dans l'une des deux bases. Un JEP permet de décrire une classe.

Tias Guns [37] propose une mesure s'inspirant de la mesure $\mathcal{X}^2(X)$ appliquée à la découverte de motifs discriminants.

Definition 16 (Mesure $\mathcal{X}^2(X)$) *Soient X un motif, $\mathcal{D}^+$ et $\mathcal{D}^-$ les bases de transactions positives et négatives, la mesure $\mathcal{X}^2$ est définie ainsi :*

$$\begin{aligned}\mathcal{X}^2(X) =& \frac{\left(supp_{\mathcal{D}^+}(X) - freq_{\mathcal{D}}(X) \cdot |\mathcal{D}^+|\right)^2}{freq_{\mathcal{D}}(X) \cdot |\mathcal{D}^+|} \\ &+ \frac{\left(supp_{\mathcal{D}^-}(X) - freq_{\mathcal{D}}(X) \cdot |\mathcal{D}^-|\right)^2}{freq_{\mathcal{D}}(X) \cdot |\mathcal{D}^-|} \\ &+ \frac{\left(|\mathcal{D}^+| - supp_{\mathcal{D}^+}(X) - \frac{|\mathcal{D}| - supp_{\mathcal{D}}(X)}{|\mathcal{D}|} \cdot |\mathcal{D}^+|\right)^2}{\frac{|\mathcal{D}| - supp_{\mathcal{D}}(X)}{|\mathcal{D}|} \cdot |\mathcal{D}^+|} \\ &+ \frac{\left(|\mathcal{D}^-| - supp_{\mathcal{D}^-}(X) - \frac{|\mathcal{D}| - supp_{\mathcal{D}}(X)}{|\mathcal{D}|} \cdot |\mathcal{D}^-|\right)^2}{\frac{|\mathcal{D}| - supp_{\mathcal{D}}(X)}{|\mathcal{D}|} \cdot |\mathcal{D}^-|}\end{aligned} \tag{2.7}$$

L'idée centrale est de déterminer les différences de distribution statistique observées pour une base donnée (positive ou négative) par rapport à la loi statistique de référence, c'est-à-dire la valeur attendue, qui correspond au support espéré du motif dans chaque base. Ce support espéré est calculé par rapport à la fréquence du motif dans la base totale multipliée par la taille de chaque base. Les deux premiers termes indiquent alors la somme des carrés des écarts pour chaque base, et leur valeur est d'autant plus grande que les écarts sont importants (c'est-à-dire que le motif possède un support dans une base qui diverge de celui prévu). Les deux termes suivants ont des valeurs qui croissent si le motif à un support qui est inférieur ou supérieur à la moitié du support dans la base totale. En effet, si la valeur du support dans une base correspond à la moitié du support global, alors la valeur du terme vaudra 0 (le motif est caractéristique des deux bases de manière équivalente), dans tous les autres cas, la valeur du terme augmentera (soit le motif est caractéristique de la base positive, soit négative).

Les motifs discriminants ont été notamment utilisé pour la classification de trajectoires de véhicules dans Lee et al. [56]. Il n'y a pas que les itemsets qui peuvent être fréquent. En effent, dans cet article, il a été démontré que les motifs séquentiels fréquents les plus discriminants étaient de bons candidats pour la génération de caractéristiques pour la construction de classifieurs car ils préservent l'ordre. Cheng et al. [21] ont proposé une approche, utilisant une exploration à l'aide d'un FP-Tree, qui extrait directement les motifs discriminants sans extraire l'ensemble des motifs fréquents. Au lieu de sélectionner les meilleurs motifs discriminants seul le plus discriminant des motifs est choisi, et un processus de sélection de caractéristiques permet de supprimer l'ensemble des transactions supportées par le motif pour réduire la base de données et continuer la recherche de motifs. L'algorithme s'arrête quand l'ensemble des transactions est retiré du FP-Tree.

La notion de motifs émergents a également été utilisée dans Métivier et al. [69]. A partir d'une base de molécules, ils cherchent à découvrir les changements structurels entre différents groupes de molécules (mutées et non mutées). Si des structures apparaissent uniquement dans le groupe des molécules mutées et non dans l'autre, alors la méthode renvoi des alertes, qui prennent la forme de concepts formels émergents. Ainsi, les auteurs

traitent dans un premier temps le jeu de données en extrayant les fragments clos fréquents de molécules. A partir de ces informations, ils vont traduire la base de molécules en description binaire, où une propriété booléenne est un fragment clos fréquent de molécule. Les motifs clos sont ensuite extraits avec LCM [89], puis les motifs émergents clos sont extraits, et enfin les motifs émergents stables sont retournés (un motif est d'autant plus stable que la probabilité que le motif conserve le même nombre de propriétés malgré le retrait d'objets dans son extension est élevée). Notre cadre méthodologique intègre l'idée d'une transformation d'objets en contexte binaire. Dans notre cas, le processus de transformation n'est pas fixe, et est piloté par l'analyste, en fonction de la connaissance qu'il cherche à extraire. Nous utilisons également les motifs émergents pour extraire des descriptions anormales de traces avec deux mesures : une mesure d'émergence normalisée ainsi que la mesure $\mathcal{X}^2$ décrite plus haut.

2.2.2 Fouille de motifs séquentiels

Les motifs séquentiels introduisent une notion d'ordre entre les itemsets. La tâche de base de la fouille de motifs séquentiels est de découvrir des sous-séquences d'items apparaissant fréquemment ce qui peut être utile pour découvrir les séquences d'achats fréquentes dans un magasin par exemple. A partir des traces unitaires, il est alors possible de créer de nouveaux scénarios intégrant la notion d'ordre, contrairement aux itemsets (ordre des processus, des localisations ou encore d'évènements de haut niveau). Après une présentation de la fouille de séquences fréquentes, nous décrirons quelques représentations condensées qui permettent une réduction du nombre de motifs sans perte d'information. Pour chaque séquence, il est possible d'assigner un itemset faisant office de dimensions, ainsi nous présentons quelques algorithmes permettant d'extraire des motifs séquentiels multidimensionnels. Nous détaillons les épisodes qui caractérisent des ensembles d'évènements proches dans le temps qui surviennent dans un ordre donné de façon récurrente, puis nous finissons par des méthodes d'extraction de motifs séquentiels avec l'appui de modèles.

2.2.2.1 Motifs séquentiels fréquents

epc_id	séquence
1	$< loc\_A \rightarrow loc\_B \rightarrow loc\_C >$
2	$< loc\_A \rightarrow loc\_B >$
3	$< loc\_A \rightarrow loc\_B \rightarrow loc\_C >$
6	$< loc\_A \rightarrow loc\_D \rightarrow loc\_E >$
7	$< loc\_A \rightarrow loc\_D >$
8	$< loc\_A \rightarrow loc\_D \rightarrow loc\_E >$

FIGURE 2.4 – Collection de traces unitaires sous la forme de séquences

Les traces unitaires étant des séquences d'évènements, il est alors intuitif de vouloir définir des scénarios dérivés des ensembles d'items pour introduire un ordre. On peut simplement définir un ordre entre les différents processus que l'on observe dans les traces. Nous définissons les concepts de séquences et sous-séquences.

Definition 17 (Séquence) *Une séquence $s =< t_1, t_2, ..., t_m >$ avec $t \subseteq \mathcal{I}$. La taille d'une séquence $|s|$ est le nombre d'itemsets dans la séquence. Une base de séquences $\mathcal{S} = \{s_1, s_2, ..., s_n\}$ est un ensemble de séquences. Chaque séquence est associée à un id.*

Exemple 8 *La table 2.4 contient 6 séquences $\mathcal{S} = \{s_1, s_2, s_3, s_4, s_5, s_6\}$. On a $s_4 =< loc_A \rightarrow loc\_B \rightarrow loc\_C >$ dont l'identifiant epc_id est 6.*

Definition 18 (Sous-séquence) *Une séquence $\alpha =< a_1, a_2, ..., a_m >$ est une sous-séquence d'une autre séquence $\beta =< b_1, b_2, ..., b_n >$ noté $\alpha \sqsubseteq \beta$ si et seulement si $\exists i_1, i_2, ..., i_m$ tels que $1 \leq i_1 < i_2 < ... < i_m \leq n$ et $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, ...,$ et $a_m \subseteq b_{i_m}$.*

Definition 19 (Support d'une séquence) *Soit S une base de séquence, le support d'une séquence α est notée :*

$$supp(\alpha) = |\{s | s \in \mathcal{S}, \alpha \sqsubseteq s\}| \tag{2.8}$$

Exemple 9 *La sous-séquence $< loc\_A \rightarrow loc\_B >$ est présente dans les séquences s_1, s_2 et s_3 et son support est de 3. Le support de $< loc\_A >$ est de 6.*

La recherche de sous-séquences fréquentes dans une base de séquences correspond à la recherche de sous-séquences dont le support est supérieur à un certain seuil, appelé seuil de support minimum. Ces sous-séquences fréquentes sont appelées des motifs séquentiels.

Definition 20 (Motif séquentiel) *Étant donné un entier positif ϵ comme un seuil de support minimum, une séquence α est appelée un motif séquentiel dans la base $\mathcal{S}$ si la séquence est contenue dans au moins ϵ séquences, c'est-à-dire si $supp(\alpha) \geq \epsilon$.*

Pour extraire les motifs séquentiels, de nombreux algorithmes ont été proposés comme PrefixSpan proposé par Pei et al. [73] qui utilise les préfixes des séquences pour construire des bases projetées pour chaque préfixe. SPAM, proposé par Ayres et al. [10], permet aussi d'extraire des motifs séquentiels et est particulièrement efficace si les séquences sont très longues. Une nouvelle stratégie de recherche en profondeur est proposée en combinant une représentation de la base de séquences en une représentation binaire verticale et une méthode de comptage du support efficace.

Ces algorithmes peuvent retourner beaucoup de motifs redondants puisqu'ils retournent chaque sous-motifs fréquents d'un motif fréquent. Pour réduire le nombre de motifs fréquents obtenus, on peut choisir d'extraire les motifs séquentiels clos.

2.2.2.2 Motifs séquentiels fermés et maximaux

Les algorithmes d'extraction de motifs séquentiels ont de bonnes performances dans les bases de séquences contenant des séquences fréquentes courtes. Cependant, quand de longues séquences sont présentes ou quand les seuils de fréquence minimum sont bas, les performances se dégradent fortement. Pour palier à cette difficulté, les motifs séquentiels clos ont été proposés [93].

Definition 21 (Motif séquentiel clos) *Une séquence α est un motif séquentiel clos si α est un motif séquentiel et que $\alpha \sqsubseteq \beta$ tel que la séquence β ne soit pas un motif séquentiel et $sup(\alpha) = sup(\beta)$.*

Le problème de l'extraction de motifs séquentiels clos est de trouver CS qui est l'ensemble des motifs séquentiels clos avec $CS \subseteq FS$, FS l'ensemble des motifs séquentiels fréquents. CloSpan utilise un ensemble de techniques d'élagage pour éviter de parcourir l'ensemble de l'espace de recherche. Le concept d'arbre lexicographique est introduit, et permet d'établir un ordre entre les séquences de telle manière qu'une séquence est supérieure à ses préfixes. L'idée centrale est d'analyser les préfixes communs et de directement explorer l'espace de recherche à partir de ceux-ci.

2.2.2.3 Motifs séquentiels multidimensionnels

Dans Pinto et al.[75], les auteurs étendent la base de séquences en base de séquences multidimensionnelles et proposent l'algorithme SeqDIM pour extraire des motifs séquentiels multidimensionnels. En reprenant les séquences vues dans la section précédente, on peut ajouter des dimensions dans la base comme par exemple *contenu* qui détermine le liquide contenu dans le produit, *couleur* la couleur du produit et *pays_magasin* qui indique le pays de vente du produit. On formalise une série de définitions pour décrire le problème de la fouille de motifs séquentiels multidimensionnels.

epc_id	contenu	couleur	pays_magasin	séquence
1	vin	gris	france	$< loc\_A \rightarrow loc\_B \rightarrow loc\_C >$
2	vin	gris	france	$< loc\_A \rightarrow loc\_B >$
3	vin	vert	france	$< loc\_A \rightarrow loc\_B \rightarrow loc\_C >$
6	cognac	blanc	allemagne	$< loc\_A \rightarrow loc\_D \rightarrow loc\_E >$
7	cognac	blanc	allemagne	$< loc\_A \rightarrow loc\_D >$
8	cognac	vert	allemagne	$< loc\_A \rightarrow loc\_D \rightarrow loc\_E >$

TABLE 2.4 – Collection de traces unitaires sous la forme de séquences multidimensionnelles

Definition 22 (Base de données multidimensionnelle) *Soit une base de séquences $\mathcal{S}$, une base de séquences multidimensionnelle a le schéma $(RID, A_1, ..., A_m, \mathcal{S})$ où RID est la clé primaire, $A_1, ..., A_m$ les dimensions.*

Definition 23 (Séquence multidimensionnelle) *Une séquence multidimensionnelle est définie par $(a_1, ..., a_m, s)$ où $a_i \in (A_i \cup \{*\})$ pour $1 \leq i \leq m$ et s une séquence avec $*$ le symbole qui permet de ne pas tenir compte d'une dimension.*

Exemple 10 *$((vin, *, france), < loc\_A, loc\_B >)$ est une séquence multidimensionnelle où la dimension couleur n'est pas prise en compte. Elle signifie que des objets ayant parcouru les localisations loc_A et loc_B ont comme contenu du vin, ont n'importe quelle couleur et ont été vendus en France.*

Definition 24 (Support d'une séquence multidimensionnelle) *Le support d'une séquence multidimensionnelle $P = (a_1, ..., a_m, s)$ correspond au nombre de tuples dans la base qui supporte P. Une séquence multidimensionnelle P supporte un tuple $t = (x_1, ..., x_m, s_t)$ dans la base de séquences multidimensionnelle si et seulement si, pour $1 \leq i \leq m$, pour chaque $a_i = x_i$ ou $a_i = *$ et $s \sqsubseteq s_t$.*

Definition 25 (Motif séquentiel multidimensionnel fréquent) *Une séquence multidimensionnelle P est appelée un motif séquentiel multidimensionnel si et seulement si $supp(P) \geq min\_sup$ avec min_sup un seuil de support minimum.*

Nous voyons que dans la Table 2.4, la séquence multidimensionnelle $((vin, *, france), < loc\_A, loc\_B >)$, qui possède un support de 3, est un motif séquentiel multidimensionnel fréquent si on fixe le min_sup à 2.

2.2.2.4 Episodes

Dans de nombreux contextes on peut collecter des suites d'évènements qui décrivent le comportement d'utilisateurs ou de systèmes. Un épisode est une collection d'évènements qui surviennent de façon rapprochée les uns des autres. L'article de Mannila et al. [66] propose d'extraire tous les épisodes fréquents dans une séquence d'évènements qui sont positionnés dans le temps à la différence des motifs séquentiels précédents où seule la notion d'ordre entre les évènements est étudiée.

Definition 26 (Evènement) *Soit E l'ensemble des types d'évènements, un évènement est une paire (A, t) où $A \in E$ est le type d'évènement et t est un entier qui indique l'emplacement dans le temps de l'évènement.*

A partir de la définition d'un évènement, on peut définir une séquence d'évènement.

Definition 27 (Séquence d'évènements) *Une séquence d'évènements s est un triplet (s, T_s, T_e) définie par : $s =< (A_1, t_1), (A_2, t_2), ..., (A_n, t_n) >$ qui est une séquence d'évènements ordonnée telle que $A_i \in E$ pour tout $i = 1, ..., n$ et $t_i < t_{i+1}$ pour tout $i = 1, ..., n-1$. T_s et T_e sont respectivement les temps de départ et d'arrivée avec $T_s \leq T_e$ pour tout $i = 1, ..., n$.*

Les épisodes peuvent être décrits comme des graphes acycliques orientés de la façon suivante :

Definition 28 (Episode) *Un épisode α est décrit par un triplet $(V, \leq, g)$ où V est un ensemble de noeuds, $\leq$ un ordre partiel dans V et $g : V \to E$ est une association pour chaque noeud avec un type d'évènement.*

On peut avoir un épisode en série quand les évènements se produisent dans un ordre précis, par exemple $\{A, B\}$ dans la figure 2.5, un épisode en parallèle quand les évènements n'ont pas de contraintes d'ordres les uns par rapport aux autres, par exemple

(C, D, E) dans la Figure 2.5, et les épisodes composites qui ne sont ni en série, ni en parallèle et qui comporte des évènements en parallèle et en série par exemple $\{(F, G)(H, I)\}$ dans la Figure 2.5. La fouille d'épisodes consiste à découvrir l'ensemble des épisodes qui couvrent une certaine fenêtre de temps de taille fixe. La fréquence d'un épisode est définie par le nombre de fenêtres que cet épisode couvre.

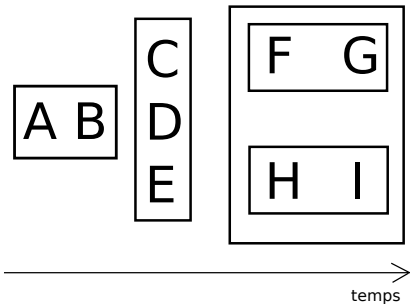


FIGURE 2.5 – Trois types d'épisodes : un épisode en série, un épisode en parallèle et un épisode composite.

2.2.3 Fouille de trajectoires

La fouille de trajectoires est un domaine de fouille de données spatio-temporelles étudié depuis peu et qui possède de nombreuses applications concernant le suivi d'objets. C'est une extension du paradigme de fouille de motifs séquentiels qui permet d'analyser les déplacements et la mobilité d'objets sous la forme de bases de données de trajectoires. Les auteurs Giannotti et al. [33] et Zheng [98] définissent les trajectoires comme des objets mobiles représentés par une séquence de différentes positions à différentes estampilles temporelles.

Une trajectoire est définie de la manière suivante : $s = p_1 \rightarrow p_2 \rightarrow ... \rightarrow p_n$ où chaque point possède des coordonnées spatio-temporelles comme $p_i = (x_i, y_i, t_i)$.

Les applications sont nombreuses : la découverte de points d'intérêt et routes ([100] [20]), pour la prédiction d'endroits ([70]) ou pour la recommandation de routes populaires pour l'utilisateur ([99]). De plus, les collections de trajectoires pouvant être volumineuses, extraire des régularités ou des groupes d'objets est un objectif important.

Une catégorie de méthodes de fouille de trajectoires s'intéresse à la découverte de groupes d'objets voyageant souvent ensemble. Ces méthodes analysent et regroupent les objets étant proches spatialement durant un ou plusieurs laps de temps. Plusieurs mesures de distances, ainsi que des contraintes temporelles plus ou moins strictes sont proposées suivant les cas d'application. Nous présenterons quatre types de motifs : les flocks, les swarms, les convois et les rGpatterns.

Gudmundsson et al. [36] définissent les *flocks* (troupeaux) comme des regroupements d'au moins m objets durant un intervalle de temps de durée min_t tel que chaque point se trouve dans un disque de rayon r qui contient les m objets, comme nous le voyons dans

le schéma (a) de la Figure 2.6. Ce type de motifs est très utile pour découvrir des lieux de réunions, des points d'intérêts, et plus généralement des objets regroupé ensembles durant un même laps de temps. Cependant, l'inconvénient est que ces objets doivent être ensemble durant un laps de temps suffisant et que la mesure de proximité (représenté par un disque) est contraignante et peu réaliste.

Les *convoys* (convois) [48] sont également des groupes d'objets proches les uns des autres durant au moins min_t estampilles temporelles consécutives, un exemple est donné par le schéma (b) de la Figure 2.6. A la différence des flocks, la forme des groupes d'objets n'est pas fixée. Un convoi est un groupe d'objets qui possède au moins m objets qui sont proches, suivant une distance e, durant au moins min_t estampilles temporelles consécutives. La limite de ce type de motifs est qu'il est rare que les objets restent ensemble suffisamment longtemps. Dans le cas de la traçabilité de produits, on constate que les objets peuvent être regroupés à un instant t, puis voyager à des instants différents (livraisons à des dates différentes) pour se retrouver stockés chez le même distributeur plus tard. Or, ces objets ne seront pas détectés dans le même convoy. D'autres motifs ont été proposés pour prendre en compte des intervalles de temps non consécutifs comme les *swarms*.

Li et al. [60] introduisent le concept de *swarms* (nuées) qui correspond à un groupe d'objets mobiles qui contient au moins m individus qui sont proches spatialement durant au moins min_t estampilles temporelles qui peuvent ne pas être consécutives. On définit un ensemble d'objets $O_{DB} = \{o_1, o_2, ..., o_z\}$ ainsi qu'un ensemble d'estampilles temporelles $T_{DB} = \{t_1, t_2, ..., t_n\}$. Ainsi, pour une liste d'objets $O = \{o_{i_1}, o_{i_2}, ..., o_{i_p}\} (O \subseteq O_{DB})$ on a un ensemble d'estampilles temporelles où les objets restent ensembles $T = \{t_{a_1}, t_{a_2}, ..., t_{a_m}\}$. Une paire (O, T) est un *swarm* si elle vérifie les contraintes suivantes : il y a au moins un cluster qui contient tous les objets de O pour chaque timestamp dans T, il doit y avoir au moins ϵ objets ($|O| \geq \epsilon$) et il doit y avoir au moins min_t timestamps ($|T| \geq min_t$).

Les rGpattern proposés par [39] permettent de décrire la diffusion d'objets dans le temps et l'espace. Un rGpattern est une liste de clusters $C^* = c_1...c_n$ dont le nombre d'objets $o \in O_{DB}$ dans chaque cluster c_i augmente ou diminue au cours du temps. Plus précisément, un rGpattern est la liste maximale de clusters d'objets qui satisfont une contrainte graduelle et une contrainte d'intégrité. Une contrainte graduelle peut être l'augmentation ou la diminution du nombre d'objets dans les clusters et la contrainte d'intégrité postule que tous les objets doivent être présents dans les clusters suivants. Deux types de rGpattern sont identifiés suivant que le nombre d'objets augmente ou diminue. Soit une liste de clusters $C^* = c_1...c_n$. C^* est un motif graduel $C^{\geq}$ dont le nombre d'objets augmente si le nombre d'intervalles de temps est supérieur à un seuil ($|C^*| \geq min_t$), si l'ensemble des objets d'un cluster c_i est inclus dans l'ensemble des objets du cluster c_{i+1} suivant ($o(c_i) \subseteq o(c_{i+1})$), si le nombre d'objets dans le dernier cluster est strictement supérieur au premier cluster ($|c_n| > |c_1|$) et enfin s'il n'existe pas de cluster tel qu'une union existe entre C^* et ce cluster ($\nexists c_m : C^* \cup c_m$). Dans le deuxième type de rGpattern, $C^{\leq}$, il faut que l'ensemble des objets du cluster suivant soit contenu dans celui du cluster courant ($o(c_i) \supseteq o(c_{i+1})$) et que la taille du cluster final soit strictement inférieure à celle du premier cluster ($|c_n| < |c_1|$).

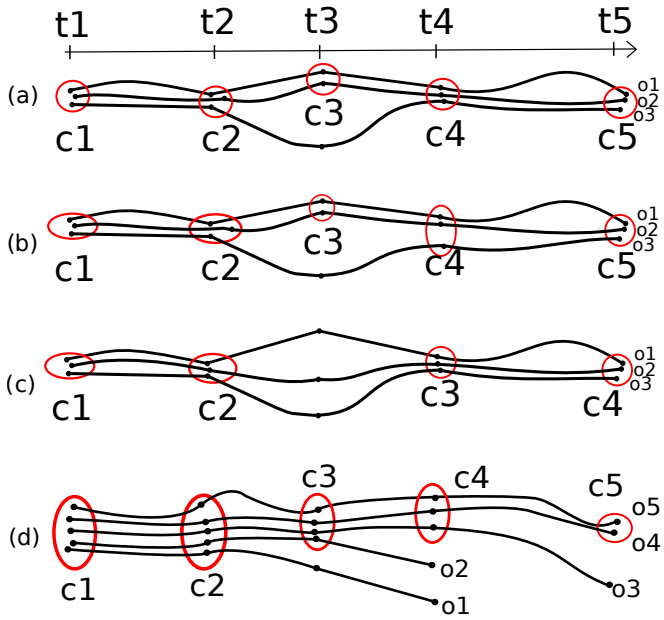


FIGURE 2.6 – Quatre types de groupes d'objets dans les trajectoires : (a) un flock, (b) un convoy, (c) un swarm et (d) un rGpattern.

2.3 Fouille de motifs pour la détection d'anomalies

Dans les sections précédentes, nous avons présenté quatre aspects de valorisation des traces unitaires disponibles : le stockage, la transformation, l'interrogation des traces unitaires, et l'extraction de connaissances avec l'utilisation de la fouille de motifs.

Un cas particulier de connaissance extraite par la fouille de données est la recherche d'anomalies dans les données. Ce cas nous intéresse particulièrement dans le domaine de l'industrie manufacturière. En effet, dans ce domaine, une anomalie peut entrainer une perte financière importante, notamment en cas de vol, de copie ou de destructions de produits.

La détection d'anomalies est un champ très actif de la fouille de données (voir l'état de l'art de Chandola et al [18]) et possède de très nombreuses applications, dans le milieu médical, celui des réseaux de capteurs ou la sécurité informatique entre autres. Hawkins [42] décrit une anomalie comme "une observation qui dévie tellement des autres qu'elle est susceptible d'être générée par un mécanisme différent". Une anomalie peut être présente dans des données pour de multiples raisons. Elle peut notamment signaler une activité suspecte, par exemple : fraude de carte de crédit ou intrusion dans des systèmes d'informations. Le but primordial de la détection d'anomalies est d'extraire des anomalies *intéressantes* pour les analystes. Extraire ces anomalies peut permettre,

par exemple, de faire des hypothèses sur les causes de celles-ci et alors permettre de mettre en place des techniques de préventions. De nombreuses méthodes prédictives pour l'extraction d'anomalies ont été proposées, exploitant différents aspects des données (angle, densité, distance, etc.). Les méthodes basées sur la distance définissent une ou des mesures de distance entre les objets à partir d'attributs numériques. Ramaswamy et al. [77] propose une méthode pour détecter des anomalies grâce aux k plus proches voisins. He et al. [44] propose une méthode basée sur le barycentre après une étape de clustering. Les objets anormaux sont les objets qui ne sont dans aucun clusters ou dans des petits clusters (distance d'un point au barycentre du plus proche cluster). Cependant, ces méthodes permettent de n'extraire qu'un type d'anomalie, or nous n'avons aucune certitude que lorsque de nouvelles traces unitaires sont captées, un modèle prédictif pourra les découvrir, puisque nous ne pouvons déterminer les informations qui entrent en jeu dans l'anormalité des traces. C'est pourquoi nous nous basons sur des méthodes descriptives, via la fouille de motifs. Nous utiliserons, en outre, des méthodes prédictives dans une autre contribution pour classifier les agents à partir de leurs traces mais non pour découvrir des anomalies.

Dans le domaine des systèmes manufacturiers, les mécanismes pouvant être responsables de l'anormalité de traces unitaires sont multiples. Ces mécanismes peuvent être humains (détournements volontaires de produits, oublis et erreurs) ou physiques (pannes, accidents).

De multiples travaux proposent d'extraire des anomalies bien définies formellement dans les traces unitaires. Dans le champ de la fouille de motifs pour la détection d'anomalies, de nombreuses techniques ont été proposées permettant d'extraire des motifs caractéristiques d'anomalies selon des sémantiques différentes. Les problèmes d'extraction d'anomalies se caractérisent selon plusieurs critères : le type de données, le type d'anomalies, si les données sont labellisées ou non, et le format de sortie des motifs. Nous introduisons un éventail de techniques pour différents langages de motifs, pouvant s'appliquer à la traçabilité de produits.

2.3.1 Systèmes ad-hoc de détection d'anomalies

Etant donné les multiples problématiques liées à la traçabilité (bruit, détournement et contrefaçon d'objets,...), de nombreux travaux se sont intéressés à la détection de types d'anomalies identifiées dans des données dont la structure est bien définie.

Staake et al. [80] proposent une architecture pour permettre la détection de contrefaçons en incluant une clé cryptée ainsi qu'un numéro d'identification dans le tag RFID. Le lecteur du tag envoie le numéro d'identification à une unité de cryptage qui génère un message crypté aléatoire et le renvoie au tag. Le tag crypte le message avec une clé puis renvoie la réponse à l'unité de cryptage. Celle-ci récupère la clé de cryptage du tag dans une base de données, crypte le message et le compare avec la réponse envoyée par le tag, ce qui permet de savoir si le tag est issu d'un objet contrefait ou non.

Lehtonen et al. [58] proposent un système d'authentification automatique basé sur les localisations et la temporalité pour calculer les probabilités de transition entre les évènements renvoyés par un produit via un tag RFID. Ils partent de l'hypothèse que le

système de production est connu et modélisé, et que des évènements rares ou anormaux peuvent être extraits en détectant des transitions improbables entre deux évènements. Si la probabilité de la transition entre deux localisations est haute, alors l'évènement généré a de grandes chances d'être produit par un produit valide. Par exemple, si un produit est détecté en Suisse puis quelques minutes plus tard au Japon, alors il y a une incohérence au niveau de la distance, la probabilité de l'évènement est basse donc la transition est considéré comme anormale.

Kerschbaum et al. [53] proposent de découvrir des contrefaçons en analysant les types d'évènements dans des traces pour y découvrir des transactions illicites entre les partenaires de la chaîne de production. Grâce à une technique qui préserve l'identité des agents logistiques, la méthode découvre les suites consécutives d'évènements, pour une trace unique, qui ne sont pas jugées conformes par les analystes (ces suites d'évènements sont conçues manuellement).

Zanetti et al. [97] proposent une méthode simple pour découvrir des produits contrefaits en analysant les traces de tags via des règles de vérification. Une trace de tag est une collection d'évènements associés à des processus de transport (envoi et réception). Les règles permettent de vérifier si un évènement de réception suit un évènement d'envoi et inversement.

De manière plus évoluée, Rao et al. [78] proposent une méthode à base de moteur de règles pour réécrire les données issues des tags RFID. Les données RFID sont vues comme un ensemble de séquences EPC, c'est-à-dire toutes les lectures d'un tag particulier (un EPC) ordonnées dans le temps. L'utilisateur peut ensuite écrire des règles avec une extension du langage de règles SQL-TS [47]. La règle est un double (A, B) qui sélectionne deux enregistrements consécutifs A et B puis applique une action exprimée en une syntaxe SQL. Par exemple on peut avoir pour un doublon (A, B) la condition $A.biz\_loc = B.biz\_loc$ AND $B.time - A.time < t\_min$ et comme action $DELETE\ B$ qui supprime l'enregistrement B s'il a la même localisation que A et si la durée entre ces deux évènements est inférieure à un seuil t_min.

2.3.2 Fouille d'ensembles infréquents et rares

Les itemsets rares ont été proposés par Szathmary [84]. L'algorithme proposé Arima permet de découvrir les motifs rares minimaux grâce à une modification d'Apriori [3] (Apriori-Rare). Si le support d'un candidat est inférieur au support minimum alors il est gardé au lieu d'être effacé. L'ensemble des motifs rares est généré à partir des motifs rares minimaux. Dans Szathmary [85], une méthode pour extraire des règles d'association rares est présentée. Ces règles se caractérisent par une confiance élevée et un support bas.

La contribution de He et al. [44] utilise les motifs fréquents pour détecter des transactions anormales. Après avoir effectué une fouille de motifs fréquents classique, un facteur d'anomalie est calculé pour chaque transaction t :

$$FPOF(t) = \frac{\sum_{X \subseteq t, X \in FPS(\mathcal{D}, min\_sup)} supp(X)}{|FPS(\mathcal{D}, min\_sup))|} \in [0,1] \tag{2.9}$$

Le facteur FPOF correspond à la proportion de transactions des motifs fréquents qui supportent t sur le nombre total des transactions faisant partie des images des motifs fréquents. Plus le facteur est proche de 0 et moins il y a de motifs fréquents qui contiennent t donc plus la transaction est considérée comme anormale. L'inconvénient est qu'il faut que l'ensemble des motifs fréquents soient extraits avant de tester chaque transaction. Giacometti et al. [32] ont proposé deux méthodes, dont une qui permet de calculer le facteur FPOF sans extraire les motifs fréquents au préalable, c'est-à-dire en opérant directement sur les paires de transactions, puis une seconde méthode qui se base sur un échantillon de motifs au lieu de l'ensemble des motifs fréquents.

2.3.3 Fouille de règles d'exception

Les règles d'exception ont été proposées par Suzuki [83]. Elles sont destinées à extraire des comportements anormaux dans certaines circonstances mais pas dans d'autres. Par exemple, si on considère la règle "utiliser une ceinture est sûr", on observe une exception avec la règle "utiliser une ceinture, pour un enfant, est risqué". C'est la spécification d'un cas particulier "pour un enfant" qui modifie la conclusion de la règle. Soit X et Y des itemsets, une règle d'exception est une paire de règles telle que :

$$\begin{cases} X & \rightarrow & Y \\ X \wedge Z & \rightarrow & \neg Y \end{cases} \tag{2.10}$$

La première règle $X \rightarrow Y$ est une règle forte et $X \wedge Z \rightarrow \neg Y$ est l'exception. Cet ensemble de règles signifie que X et Y sont souvent observés ensemble, mais que si X est en conjonction avec Z, alors X et Y ne sont plus observés ensemble. Autrement dit, cette paire de règles permet de capturer les comportements où l'interaction entre deux évènements X et Z altèrent le comportement normal (c'est-à-dire la présence d'un évènement Y).

2.3.4 Fouille de sous-trajectoires anormales

La découverte d'anomalies dans les trajectoires possède plusieurs champs : soit la découverte d'une trajectoire ou de portions de trajectoires significativement différentes d'autres trajectoires selon une métrique (détours de taxi, ou changement de routes brutal) ; soit la découverte d'évènements anormaux en utilisant des collections de trajectoires (accidents, manifestations, évènements sportifs, ...).

Chen et al. [19] proposent une méthode de détection de portions de trajectoires anormales en temps réel. Premièrement, une base de trajectoires est constituée. Une trajectoire est testée en fonction de n points entrants, n étant la taille de la fenêtre. Si la séquence des points possède un support inférieur à un certain seuil θ, c'est-à-dire que

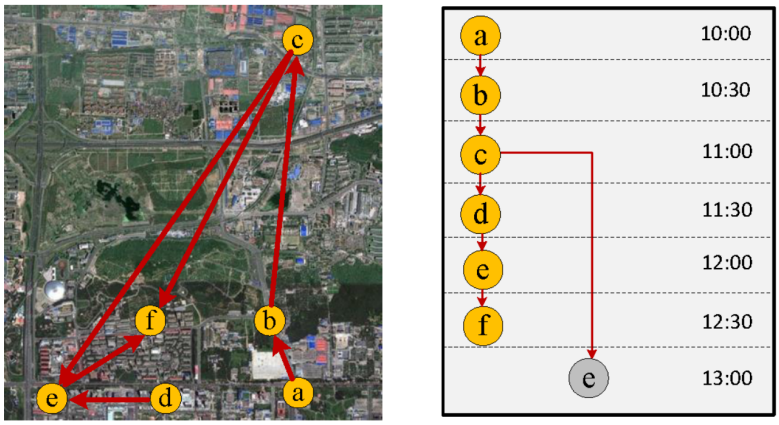


FIGURE 2.7 – Exemple d'un sous-arbre fréquent anormal issu de [61]

le nombre de trajectoires qui contiennent cette séquence de points est inférieur à θ, alors cette séquence est considérée comme anormale.

Liu et al. [61] proposent une méthode de détection d'évènements anormaux dans des collections de traces de taxis. Ils divisent l'espace (une ville) entre plusieurs régions automatiquement et calculent trois caractéristiques pour chaque lien et pour chaque unité de temps : le nombre de véhicules qui transitent, le nombre de véhicules qui entrent dans la région de destination, le nombre de véhicules qui partent de la région d'origine. L'idée centrale est de calculer l'évolution de chaque caractéristique au cours du temps et de calculer la distorsion minimum pour chaque caractéristique. Ces dimensions peuvent être projetées dans un espace en 3D et ils cherchent les points extrêmes (grâce à la distance de Mahalanobis) pour découvrir les anomalies. La Figure 2.7 montre l'exemple d'une anomalie, sous la forme d'un sous-arbre fréquent spatio-temporel. Chaque noeud est un point d'intérêt, et chaque transition correspond à des passages fréquents de taxis entre deux points. L'exemple démontre qu'il y a beaucoup plus de véhicules qui circulent que dans les autres banlieues, ce qui démontre un besoin de transports en commun. Les auteurs ont d'ailleurs remarqué que dans cette zone, un métro allait être construit.

2.3.5 Fouille de sous-séquences anormales

L'objectif de la recherche de sous-séquences anormales dans une séquence d'évènements est de découvrir des portions de cette séquence anormales ou uniques. Gwadera et al. [38] veulent découvrir si un certain nombre d'occurrences d'une sous-séquence peut être indicatif d'une activité suspecte. Soit une fenêtre de temps donnée, les auteurs déterminent le comportement normal d'un épisode en calculant le nombre de fenêtres qui contiennent au moins une occurrence d'un épisode. Si un épisode s'écarte de ce comportement moyen, alors il est suspect et lève une alerte.

Lorsque nous avons une chaîne de caractères, par exemple une séquence d'évènements, nous pouvons utiliser le concept de plus petite sous-chaîne unique comme moyen

de découvrir des évènements anormaux. La plus petite sous-chaîne unique ou encore SUS (Shortest Unique Substring) correspond à une sous chaîne présente dans la chaîne dont elle est issue et dont la longueur ne peut pas être réduite sans qu'elle ne soit plus unique. Elle a été étudiée dans Haubold et al. [41] pour permettre la détection de régions uniques dans une séquence ADN sans faire une étape de comparaison avec une base existante de séquences. Cette méthode est adaptée quand le nombre de symboles utilisés est faible, car il entraine de nombreuses redondances et les parties uniques sont plus difficilement détectables. L'article de Jian Pei et al.[74] présente les SUSQ (Shortest Unique Substring Queries) étudiées également dans Tsuruta et al. [88] et Ileri et al. [9]. Ces requêtes permettent de découvrir, à partir d'une position p et d'une chaîne S, la SUS qui est unique dans S contenant la position p.

2.3.6 Fouille de motifs séquentiels à l'aide de modèles experts

L'idée d'utiliser des modèles pour découvrir des séquences intéressantes a été étudié par Jaroszewicz et al. [45]. Dans cette étude, il utilise des modèles de Markov cachés qui permettent de décrire des systèmes de transitions entre états avec une temporalité discrétisée. En sortie, un modèle de Markov caché produit un ensemble de symboles et il est possible de calculer la probabilité que ce modèle produise une séquence de symboles donnée notée $Pr^{HMM}\{\mathcal{O}\}$ où $\mathcal{O}$ est une séquence de symboles. La probabilité d'observer une séquence $\mathcal{O}$ dans les données est quant à elle égale à la fréquence de cette séquence de symboles dans une base D, soit le pourcentage de séquences qui possèdent comme préfixe $\mathcal{O}$ dans D notée $Pr^{D}\{\mathcal{O}\}$. Pour finir, les auteurs calculent la mesure d'intérêt d'une séquence en faisant la différence entre la probabilité obtenue à partir du modèle de Markov caché et la fréquence soit $\mathcal{I}(\mathcal{O}) = |Pr^{D}\{\mathcal{O}\} - Pr^{HMM}\{\mathcal{O}\}|$.

Sun et al. [81] proposent une méthode pour découvrir les séquences de symboles les plus anormales par rapport à un modèle calculé à partir d'une base de séquences. Le modèle utilisé est un PST (Probabilitistic Suffix Tree) qui est une représentation compacte d'une chaîne de Markov d'ordre variable, et qui utilise un arbre de suffixe comme structure d'index. Dans une chaîne de Markov d'ordre variable, le nombre de variables aléatoires évolue pour calculer la probabilité d'observer l'état suivant. Le PST encode les probabilités d'observer un symbole donné après une séquence de symboles (un préfixe) et détermine alors les séquences possibles que le système doit générer. Les auteurs utilisent cet arbre pour déterminer si une séquence correspond bien à une séquence possible (c'est-à-dire si elle a une probabilité forte de réalisation calculée à partir du PST) par rapport à une séquence aléatoire. Une séquence est anormale si elle est improbable et si sa probabilité d'apparition est proche d'une séquence aléatoire. Cette méthode a été étendue par Low-Kam et al. [62] qui proposent deux améliorations : un critère d'élagage supplémentaire qui ajoute uniquement les noeuds qui diffèrent statistiquement de leurs parents et qui permet de réduire la taille de l'arbre, et une inégalité pour la concentration de la mesure de similarité qui permet de mieux trier les anomalies.

Ces méthodes se basent sur une structure de données qui exprime ce qui est attendu en intégrant la dimension temporelle (transitions entre états avec les HMM, et arêtes entre les noeuds de l'arbre de préfixe). Nous souhaitons utiliser l'idée de fixer ce qui

est normal et attendu par un modèle et intégrer pleinement ce modèle dans le processus général de fouille de données. Cependant, nous constatons que l'utilisation de ces modèles reste restreint au calcul d'une mesure d'anormalité fixe. Nous proposons une méthode qui permet d'agir sur le modèle et également plusieurs mesures adaptées au modèle et aux données qui serviront de support au calcul de l'anormalité d'un objet.

2.3.7 Fouille d'anomalies contextualisées

Le champ des anomalies contextuelles est un champ récent qui s'attache à chercher les contextes où des objets sont anormaux alors qu'ils ne le sont pas dans la base de données globale. Chaque instance des données possède deux ensembles d'attributs : les attributs contextuels qui sont utilisés pour déterminer le *contexte* de l'instance (par exemple, la position dans l'espace, dans le temps ou d'autres attributs numériques) ; les attributs de comportement qui permettent de définir les caractéristiques non contextuelles qui servent à découvrir un comportement anormal dans un contexte spécifique. Dans notre deuxième contribution, nous utilisons ce formalisme en décrivant ce que nous appelons les trajectoires comme les attributs comportementaux, et les descriptions, comme les attributs contextuels. Les travaux de Dang et al. [24] et Duan et al. [26] permettent d'extraire les objets anormaux et leurs contextes (aspects) dont les attributs sont numériques. L'idée centrale est d'extraire les ensembles d'attributs qui permettent de définir des sous-espaces numériques possédant une haute densité d'objets anormaux, alors que ces objets ne sont pas anormaux dans l'espace de tous les attributs. Tang et al. [86] proposent d'extraire des motifs multidimensionnels dans des bases de données catégorielles qui vont décrire des anomalies. Ces anomalies prennent la forme d'une paire d'ensembles multidimensionnels (dans le cas d'application, ils décrivent des groupes de personnes selon leur âge, leur dernier diplôme, et leur situation amoureuse), tel que par exemple $((*, bac, célibataire), (*, bac, divorcé))$ dont le degré est la couverture du groupe de référence (1 élément de la paire) sur celle du groupe anormal, ici il est de 5.7. Le groupe de référence est un ensemble de 34 personnes, tous âges confondus, ayant le baccalauréat et étant célibataire et le groupe anormal est un groupe de 6 personnes ayant également le baccalauréat mais étant divorcés. Une série de travaux d'Angiulli et al. [5][6][4] propose de s'appuyer sur la programmation logique et une base de connaissance sous forme de règles pour décrire les objets anormaux. Une connaissance est exprimée sous la forme de règles, et un ensemble d'exemples classés en positif ou négatif est disponible. La technique proposée permet de fournir des explications des anomalies, en cherchant à découvrir les prédicats qui vont décrire uniquement les exemples négatifs. Dang et al. [23] propose une méthode permettant d'explorer et de caractériser des anomalies locales dans des jeux de données à haute dimension. Elle permet non seulement d'extraire les objets anormaux, triés par degré d'anormalités, mais aussi de découvrir un ensemble restreint de caractéristiques discriminantes expliquant pourquoi une anomalie est exceptionnelle par rapport aux autres motifs normaux. La méthode est appliquée sur une base d'images de taille 32×30 pixels étant chacun des vecteurs de 960 caractéristiques. Composé de visages principalement neutres (expression neutre, pas d'accessoires), la méthode affecte un score plus élevé pour les images possédant des caractéristiques divergents fortement

des autres visages : dans leur exemple, le visage ayant le score le plus élevé possède des lunettes de soleil et le second est un visage souriant.

Les anomalies collectives, présentée par Song et al. [79], modélisent une collection d'instances qui sont anormales, prises ensembles, par rapport à l'ensemble de la base de données, mais où chaque instance n'est pas une anomalie par elle-même. Ce travail permet de découvrir les ensembles, ou séquences d'objets ou d'évènements, qui sont responsables conjointement d'une anomalie.

2.4 Conclusion

Dans cet état de l'art nous avons pu découvrir un ensemble de travaux permettant de valoriser des traces unitaires, et plus particulièrement des traces issues de systèmes manufacturiers, traduites en différents langages : ensembles de propriétés, séquences et épisodes. Trois possibilités de valorisations ont été montrées : interrogation des traces unitaires, traduction des traces en un langage de motifs pour la découverte d'évènements et enfin la découverte d'anomalies dans ces traces. Les travaux importants sont résumés dans le Tableau 2.8. Nous observons qu'aucun travail ne permet d'offrir un cadre générique pour la fouille de traces unitaires depuis la sélection, la transformation en différents langages de motifs et la fouille de traces unitaires pour la description ou l'extraction d'anomalies. Le domaine du système manufacturier permet de surcroît de faire intervenir activement la connaissance experte dans la détermination des comportements normaux utiles à la détection des comportements déviants.

Dans cette thèse, nous proposons un cadre méthodologique permettant la manipulation des traces unitaires pour définir des scénarios de fouille. De nombreux langages sont disponibles pour décrire chaque trace unitaire : des propriétés booléennes, des intervalles numériques, des séquences, des graphes, etc. Nous nous sommes focalisés sur une étape de transformations des traces unitaires en une structure de données bien connue qui sont les itemsets. L'idée est de proposer aux experts une méthode simple et efficace en décrivant la connaissance contenue dans les traces en propriétés booléennes. Dans notre cas, l'extraction d'itemsets clos est effectué par un algorithme efficace (CHARM [96]) pour découvrir des ensembles de propriétés booléennes fréquents et non redondants. Ainsi, à partir de transformations des données initiales en contextes de fouille différents, nous pouvons extraire de la connaissance variée des traces unitaires.

Nous démontrons comment les différentes étapes (interrogation, sélection, transformation, fouille) peuvent être agencées pour extraire des informations intéressantes en faisant interagir les traces, les trajectoires (traces transformées en un langage de motifs), les motifs extraits et le modèle de filière (connaissance experte sous forme de graphe).

Pour l'extraction d'anomalies nous proposerons une méthode de fouille de motifs intelligibles caractéristiques d'anomalies basés sur le concept de motifs émergents [25] et faisant intervenir la connaissance experte pour déterminer l'anormalité d'une trace, que nous étendons par la suite avec la découverte de couples de motifs discriminants définissant deux phénomènes semblables mais qui diffèrent sensiblement, décrivant une anomalie.

Enfin, nous proposerons une méthode originale pour la découverte d'agents fraudeurs dans les traces de comportement produites par des agents logistiques (distributeurs par exemple). Dans ce scénario, nous avons démontré qu'après avoir entrainé un classifieur sur ce type de données, nous pouvons découvrir avec une haute précision la personne qui a produit ces traces. En revanche, dans le cas où les agents possèdent plusieurs identités (pour détourner des produits après avoir été exclus du système par exemple), sans que l'on sache lesquelles, la précision chute fortement. A notre connaissance, aucun travail ne s'attaque à la découverte de ces usurpations d'identités dans les traces.

Objectif	Langage	Type de motifs	Articles
Descriptif	Itemsets	Fréquents	Agrawal et al. [3]
		Clos et maximaux	Pasquier et al. [72], Zaki et al. [96]
		Émergents	Dong et al. [25], Guns [37]
		Avec modèles	Flouvat et al. [29], Jaroszewicz et al. [46]
	Motifs séquentiels	Fréquents	Pei et al. [73]
		Clos et maximaux	Yan et al. [93]
		Multidimensionnels	Pinto et al. [75]
		Avec modèles	Sun et al. [81],Jaroszewicz et al. [45]
	Trajectoires	Fréquentes	Gianotti et al. [33], Zheng [98]
		Point d'intérêts	Zheng et al. [100] Zaiben et al. [20] Monreale et al. [70]
		Clusters temporels	Gudmundsson et al. (flock) [36] , Jeung et al. (convoy) [48], Li et al. (swarm) [60] , Hai et al. (rgpattern) [39]
	Workflows	WF-Nets	Aalst et al. [1]
Anomalies	Itemsets	Infréquents et rares	Szathmary et al. [84]
		Règles d'exception	Suzuki et al. [82]
	Séquences	Anormales	Gwadera et al. [38] Haubold et al. [41] Pei et al. [74] Tsuruta et al. [88] Ileri et al. [9]
	Trajectoires	Anormales	Chen et al. [19], Liu et al. [61]
	Variées	Contextuelle	Duan et al. [26] Tang etal. [86] Angiulli et al. [5][6][4] Dang et al. [24][23]
		Collective	Song et al. [79]

FIGURE 2.8 – Sélection de travaux issus de l'état de l'art de la fouille de données

Chapitre 3

Un cadre méthodologique pour la fouille de traces unitaires

Dans l'état de l'art, nous avons décrit de nombreuses techniques de fouilles de motifs, notamment sur des données ensemblistes. Si des techniques très efficaces existent pour extraire des motifs de ce type, la valeur informative des motifs est très basse si les propriétés encodées ne permettent pas de décrire des co-occurences d'évènements normaux ou anormaux. Par exemple, si nous encodons une propriété par gamme de produits, mais que les anomalies concernent des groupes de produits ayant des prix de vente semblables, alors la fouille de motifs ne donnera rien, et il sera préférable de discrétiser le prix de vente des produits pour permettre de caractériser les anomalies. Nous allons démontrer l'utilité de la fouille de motifs dans divers scénarios et comment l'étape de transformation des données captées, en utilisant la connaissance experte, est cruciale pour construire des contextes de fouille dont les propriétés permettront de décrire intelligemment les "comportements" des produits. De nombreuses problématiques émergent comme la volumétrie des traces, les contraintes physiques introduisant de l'incertitude dans la récupération de données de qualité et la multiplicité des comportements possibles.

En revanche, la grande richesse des informations disponibles, tant au niveau des données de traçabilité captées, des données captables attendues, ou de la structure du système manufacturier, permet de s'intéresser à l'intégration de la connaissance experte à différentes étapes du processus de valorisation des traces. Les connaissances expertes et du domaine pourront servir à la génération de nouvelles connaissances de haut niveau (se basant sur les coordonnées géographiques, la temporalité, les informations sémantiques assignées à chaque produit ou chaque localisation,...) pour former des *contextes de fouilles* à partir desquels on peut appliquer des solveurs. Si l'on veut extraire les ensembles de point d'intérêts fréquemment visités par les produits, on pourra alors construire un contexte de fouille comportant une propriété booléenne si un point d'intérêt est visité par le produit, puis appliquer un algorithme de fouille de motifs ensemblistes. D'autre part, la connaissance experte du système manufacturier peut être construite sous la forme d'un réseau de localisations connectées par des flux de produits. La construction d'un tel modèle est spécifique aux domaines d'application (on peut choisir de l'inférer

à partir des traces unitaires, ou de le construire manuellement). La connaissance peut aussi être utilisée pour assister la fouille de données directement comme pour la fouille d'itemset avec Jaroszewicz et al [46] [45] ou Flouvat et al. [29].

Nous proposons un cadre méthodologique permettant la découverte de connaissance à partir de traces unitaires en incluant une dimension générique grâce à la diversité des langages de motifs utilisables (contextes de fouille). La question d'un codage intelligent et l'injection de connaissances expertes et/ou du domaine directement dans les propriétés constituant les instances de la base de données n'ont pas fait l'objet de travaux dans le domaine de la traçabilité de produits. Nous apportons alors une contribution en dégageant, à partir d'un processus global et générique, l'ensemble des degrés de liberté pour la valorisation des traces : sélection et interrogation des traces, calcul des propriétés à partir des traces pour la génération de contextes de fouille, interrogation et fouille des contextes, interrogation des motifs issus de la fouille, interrogation, visualisation et fouille du modèle de filière et enfin les articulations possibles entre ces différentes tâches.

3.1 Le cas de l'analyse d'une filière manufacturière

Le cadre présenté dans ce manuscrit correspond à l'analyse des données de traçabilité produits dans le contexte de la distribution d'objets physiques depuis un fabricant à un client. Après avoir présenté la topologie du contexte manufacturier et les différents acteurs en détail, nous allons décrire les choix et hypothèses d'abstraction que nous avons faits pour construire un cadre méthodologique d'analyse de traces unitaires de produits manufacturés. Enfin, nous décrivons différents scénarios pour démontrer la pertinence de ce cadre méthodologique et les degrés de liberté qu'il fournit.

3.1.1 Présentation générale du contexte manufacturier

Le contexte manufacturier possèdent de nombreux types de connaissances. D'une part, le réseau de localisations par lequel passe les objets possède une topologie propre. Ces localisations, que l'on appelle également sites, ont des attributs divers (nom du site, coordonnées spatiales, type du site, entreprise propriétaire du site,...), des fonctions diverses (création, décoration, destruction) desquelles découlent des processus différents et donc des "comportements" des produits différents. La Figure 3.1 présente un panorama détaillé de l'ensemble du processus typique d'un système manufacturier utilisant le vocabulaire du protocole EPCIS. Chaque agent logistique est caractérisé par un rectangle bleu (producteur, logisticien, intermédiaire, distributeur, consommateur). Chaque agent va procéder à des actions typiques, décrites sous la forme d'évènements, qui peuvent être commune à tous (conditionnement, stockage, envoi, réception,etc.) ou spécifique (comme la décoration ou la vente). Chaque évènement possède un type, présenté sous la forme d'étiquettes blanches (commissioning, packing, storing, etc.) et est enregistrée sous une base de données EPCIS [5]. La séquence de ces actions constitue le cycle de vie typique

5. La base de données utilisée dans le cadre du projet est MongoDB, `https://www.mongodb.com/fr`

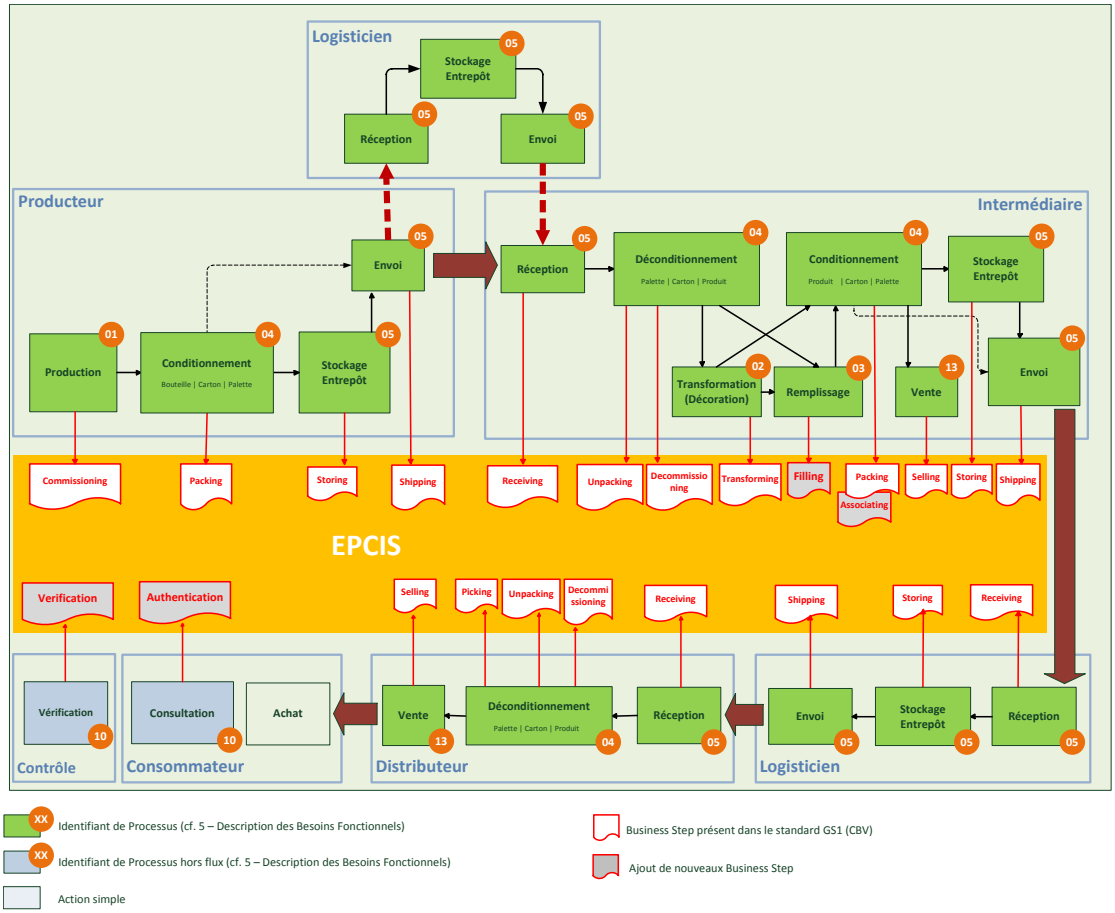


FIGURE 3.1 – Processus général d'un système manufacturier utilisant le protocole EPCIS

d'un produit que l'on nomme trace unitaire. Les fonctions et intérêts spécifiques des différents acteurs de la chaîne logistique (propriétaire d'un ou plusieurs sites) influent alors différemment sur le cycle de vie du produit dont la trace exprime les données captées du système en fonctionnement. Le réseau de localisation est connu par certains acteurs, en totalité ou partiellement, et il est possible de savoir quels sont les localisations de départ et d'arrivée lorsqu'un ensemble de produits est déplacé d'une localisation à une autre. Cette connaissance des flux d'objets est une information attendue, c'est-à-dire que c'est une connaissance du système lorsque celui ci est dans un fonctionnement jugé conforme et en accord avec le processus métier permettant l'acheminement des produits à des acteurs connus du système et autorisés à distribuer les produits dans des zones géographiques correspondantes. En effet, pour chaque acteur qui transforme ou distribue le produit, il est attendu qu'il le fasse selon des contraintes industrielles bien connues et fixées notamment par contrat et par des lois comme par exemple la distribution d'un certain type de produits dans des zones et pays spécifiques à des prix contrôlés.

Or, premièrement, la nature même du système comporte des contraintes physiques

entrainant des accidents ou des retards de livraison par exemple et modifiant le comportement attendu. Deuxièmement, certains acteurs, généralement peu ou pas connus des acteurs qui fabriquent les produits, dérogent aux limitations de vente et de distributions des produits par intérêt financier.

3.1.2 Abstractions et hypothèses à partir d'un cas réel

Précédemment, nous avons décrit le contexte manufacturier et nous pouvons observer deux types de données : les données captables (données attendues) et les données captées (données issues des capteurs du système manufacturier). Ces deux types de données possèdent des structures et des fonctions différentes que l'on propose de décrire dans les sous-sections suivantes. Nous développons les abstractions et les hypothèses faites pour permettre ensuite une valorisation des traces unitaires.

Données captables : le modèle de filière Les données que l'on nomme captables sont les données, ou un aspect des données, que les experts attendent en sortie du système en fonctionnement. Ces données sont potentiellement très riches puisqu'elles peuvent porter sur les localisations, les transitions entre ces localisations, les attributs des acteurs, les types de processus attendus. Elles sont issues d'un procesus qui peut être automatique, semi-automatique ou manuel. Dans le cas des systèmes manufacturiers plusieurs modèles peuvent être imaginés : graphe des sites par lesquels passent les produits, séquences des processus courants appliqués aux produits [90], évolutions des attributs d'un produit au court du temps, règles métiers, etc. L'ensemble de ces modèles vont nous permettre de valoriser les traces en calculant des propriétés informatives en elles mêmes et pouvant être interrogées mais également pouvant être le support d'algorithmes de fouille de données. A un autre niveau, ces modèles peuvent aussi permettre de déterminer ce qui est considéré comme attendu, ou normal, par les experts, et ainsi classer et pondérer dans les données captées par le système, les traces/évènements/propriétés normales et anormales.

Nous proposons, au sein de notre cadre méthodologique, d'évaluer le degré d'anomalies des traces unitaires avec le modèle de filière. Notre but est d'exprimer des fonctions telles que, pour une trace t et un modèle M, elles renvoient une classe telle que $M(t) \rightarrow \{+, -\}$ qui vont déterminer si une trace est positive (normale) ou négative (anormale) ce qui permettra d'utiliser des méthodes de fouille de motifs discriminants afin de caractériser la collection de traces négatives, en vue de décrire des anomalies. Dans ce manuscrit, nous exprimons la connaissance experte sous la forme d'un graphe orienté. Chaque nœud est un site, lieu d'actions effectuées sur un produit, et chaque arête une transition autorisée, ou attendue, par l'expert entre deux sites (action de transport des produits). A partir de cet exemple simple, nous proposons des méthodes de découverte d'anomalies dans des traces unitaires testées sur des données réelles.

Données captées : le protocole EPCIS Les données captées sont les données issues des capteurs installés sur la chaine de distribution pour enregistrer les différentes actions effectuées sur les objets. Les données captées obéissent à des contraintes fortes tant au niveau de la structure de données, que des domaines de valeurs que peuvent prendre les attributs. Les données sont générées en temps réel, sous forme d'enregistre-

ments, et possèdent toutes la même structure, sous forme de tuples d'attributs. Chaque attribut possèdent un domaine de valeurs et un type : numérique, catégoriel, position spatiale, intervalle, etc. Les domaines de valeurs des attributs peuvent être très grand, comme le temps, la position géographique, le nom du lieu, ... mais aussi restreint comme par exemple le type de l'action effectuée sur l'objet qui ne dépasse pas les dizaines de valeurs possibles. La notion de séquentialité est également importante. En effet, les enregistrements sont captées à des temps et positions différents, et les processus varient suivant les processus précédents effectués sur les objets. A cela, s'ajoutent de multiples contraintes physiques, humaines et environnementales : intégrité et disponibilité des capteurs, oublie ou erreurs de scannage, accidents de transport, évènements météorologiques ralentissant ou bloquant le transport ou la production de marchandise. Cet ensemble de contraintes provoque des erreurs d'enregistrement et des données manquantes. Les systèmes de production et de distributions des produits impliquent également le fait que chaque lieu de distribution a une confiance et un taux d'erreur différent : en effet, si plusieurs sites (usine et entrepôt de stockage) appartiennent à la même entreprise, les informations sur les taux d'erreurs, les équipements en capteur sont susceptibles de mieux être connus que d'autres sites appartenant à d'autres entreprises collaboratrices. L'ensemble de ces informations correspondent à la connaissance experte des systèmes manufacturiers. Les processus appliqués aux objets, les rôles des acteurs, les informations descriptives des produits par exemple appartiennent aux connaissances expertes qui décrivent le système manufacturier en fonctionnement. Les erreurs de fonctionnement, les accidents, les détournements de produits appartiennent à la connaissance experte du système en défaillance que ce soit à cause de facteurs environnementaux ou d'actions intentionnelles de fraude de produits.

3.2 Formalisation des données de traçabilité

Les traces unitaires sont des séquences d'évènements EPCIS [6] (Electronic Product Code Information Services) qui contiennent la séquence d'évènements captées tout au long du cycle de vie d'un produit. Le protocole EPCIS fournit un modèle qui permet de décrire des enregistrements qui proviennent de scanneurs et chargé de représenter différentes actions au sein d'un système manufacturier. Notre système d'analyse est centré sur les produits, nous regroupons ces enregistrements par produit et nous les trions en fonction du temps. Nous appelons *trace unitaire* la séquence des enregistrements captés par un système donné lors du déplacement d'un produit. A chaque objet correspond une et une seule trace.

Definition 29 (Enregistrement) *Soit un ensemble d'attributs* $A = \{A_1, ..., A_n\}$ *numériques ou catégoriels. Un enregistrement* $r \in R$ *est un n-uplet* $r = (a_1, ..., a_n)$ *avec* $a_i \in dom(A_i)$.

Definition 30 (Trace unitaire) *On note* $t = \langle r_1, ..., r_k \rangle$ *une trace unitaire avec* $r_i \in R$.

6. http://www.gs1.org/sites/default/files/docs/epc/epcis_1_1-standard-20140520.pdf

Une trace unitaire indique la suite d'évènements captés lors du cycle de vie d'un produit. Une collection de traces est notée $\mathcal{T}$.

La trace unitaire $t \in \mathcal{T}$ d'un objet dénote son comportement tout au long de son déplacement dans un réseau de sites. La trace unitaire correspond aux comportements *captés*. Cette notion est importante puisqu'un enregistrement capté l'est sur un site donné du système manufacturier. Or ce site peut ne pas être connu des experts ou pris en compte dans le système d'analyse, l'enregistrement apparaitra comme étant capté sur un lieu inconnu. Également, l'ordre du captage des enregistrements est connu et constitue l'ordre des actions nécessaires pour créer, distribuer et vendre un produit de façon attendu par les experts. Si l'ordre change, possède des actions non prévues ou manquantes, alors le processus global est perturbé. Cependant, il est nécessaire de déterminer, via un modèle, ce qui est prévu ou non, pour déterminer les sections anormales d'une trace et le taux d'anormalité (par exemple le nombre d'enregistrements manqués). A la manière de Rao et al. [78], il est possible de construire un ensemble de règles pour déterminer si deux enregistrements consécutifs vérifient un certain nombre de contraintes (par exemple le temps écoulé ou la distance parcourue entre les deux enregistrements).

Les traces unitaires sont produites en grand nombre (plusieurs centaines de milliers par jour dans le cadre de notre projet) et portent sur de longues périodes de temps ainsi que sur des espaces géographiques potentiellement très grand (les sites peuvent être positionnés dans n'importe quel pays). De plus, les produits peuvent voyager et être destinés à des marchés spécifiques et dissemblables par rapport aux profils des clients ciblés pour la vente de ces produits. Pour toutes ces raisons, l'analyse des traces de produits devra porter sur des sous-ensembles de traces unitaires spécifiques à un questionnement précis. Cette sélection des traces unitaires peut se faire au début du processus d'interrogation ou de fouille de traces unitaires, ou bien en fonction des résultats obtenus par les processus d'analyse.

3.3 Découverte de motifs appliquée à l'analyse de collections de traces unitaires

Nous décrivons formellement chaque étape du cadre méthodologique décrit par la Figure 3.2 et présentons des exemples issus de cas d'étude manufacturiers. Premièrement, nous exposerons l'étape de sélection des traces unitaires, puis l'étape cruciale de transformations de cette collection de traces en trajectoires contextualisées de produits. Ces contextes de fouille peuvent être traités de plusieurs façons différentes : soit en sélectionnant celles qui possèdent certains évènements, soit en appliquant une fouille de motifs sans le modèle de filière (pour extraire des co-occurrences fréquentes) ou bien avec le modèle de filière pour déterminer les motifs caractéristiques d'anomalies.

3.3.1 Sélection d'un ensemble de traces unitaires

Les traces unitaires peuvent être organisées en collection de traces obtenues via des requêtes pour cibler les données à analyser. Dans le cas de l'analyse de données des sys-

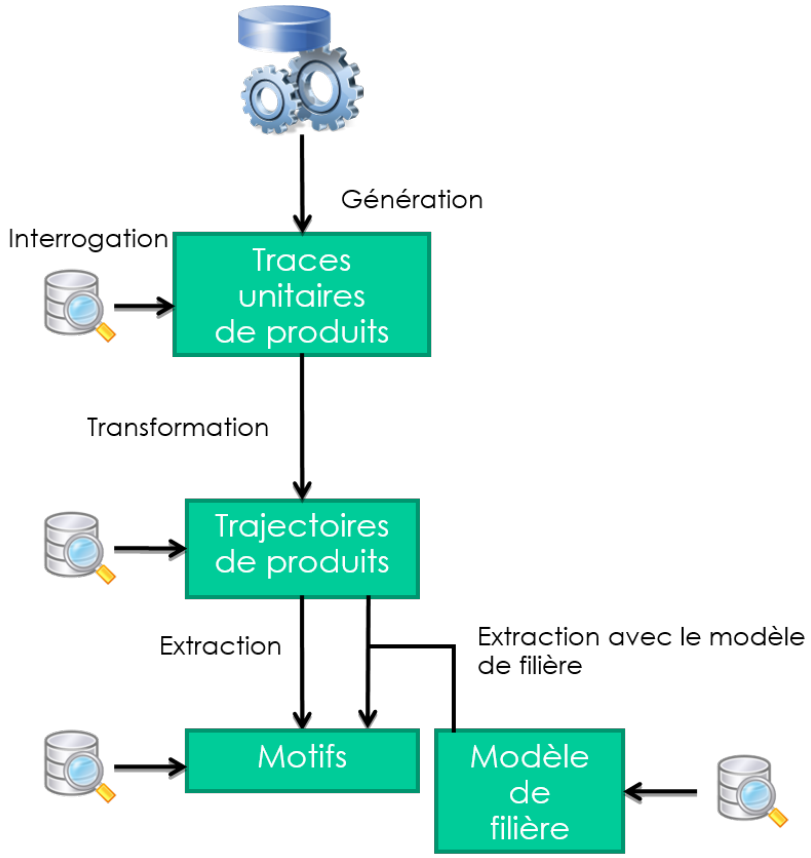


FIGURE 3.2 – Schéma des processus présents au sein de notre méthodologique

tèmes manufacturiers, il est fréquent, et pertinent, de ne s'intéresser qu'à une collection de traces unitaires issues de produits ayant circulés sur un marché précis, vendus dans des zones géographiques spécifiques (par exemple un continent ou un pays) et durant un temps donné (par exemple les produits construits, acheminés puis vendus durant un seul trimestre). Ces sélections de traces unitaires sont possibles via des systèmes d'interrogation classiques tel que SQL ou même via des expressions régulières sur les données brutes.

Le processus de sélection correspond à une requête $q \in Q$ qui créée un nouvelle collection de traces unitaires à partir de $\mathcal{T}$. Ces requêtes permettent de sélectionner un sous-ensemble de traces unitaires à partir de contraintes sur le domaine de valeurs d'un ou plusieurs attributs des évènements des traces unitaires. Les sélections peuvent aussi se faire à l'aide du modèle de filière ou via les résultats de la transformation des données durant l'étape de filtrage.

3.3.2 Codage des traces unitaires en contextes de fouille

Cette étape de notre méthodologie s'attache à traduire l'ensemble des traces unitaires en un ensemble d'instances exprimées dans un type de données associé à un langage de motifs $\mathcal{L}$ quelconque. Nous nommons chaque instance crée par un processus de transformation, une *trajectoire* :

Definition 31 (Trajectoire contextualisée) *Soit $t \in \mathcal{T}$ une trace unitaire. La trajectoire de t est une séquence de sites $trajectory(t) = \langle v_1, ..., v_n \rangle$ où $v_i \in V$ et V est un ensemble de sites. La description de t est un ensemble d'items $description(t) \subseteq \mathcal{I}$. Une paire $(trajectory(t), description(t)), \forall t \in \mathcal{T}$ est nommée une trajectoire contextualisée.*

epc_id	$trajectory(t)$	$description(t)$
1	$\langle loc\_A, loc\_B, loc\_C \rangle$	$\{vente, nombre\_stockage\_2, passage\_loc\_A, passage\_loc\_B, passage\_loc\_C\}$
2	$\langle loc\_A, loc\_B \rangle$	$\{nombre\_stockage\_2, passage\_loc\_A, passage\_loc\_B\}$
3	$\langle loc\_A, loc\_B, loc\_C \rangle$	$\{vente, nombre\_stockage\_2, passage\_loc\_A, passage\_loc\_B, passage\_loc\_C\}$

TABLE 3.1 – Trois trajectoires contextualisées

La fonction $trajectory(t)$ représente les données de comportement. Dans notre cas, nous exprimons les données de comportement par une séquence de sites telle que $\langle s_1, ..., s_k \rangle$ où s_i est le i-ème site, et k le nombre total de sites de la trajectoire. Dans l'exemple donné dans la table 3.1, les trajectoires sont présentées dans la deuxième colonne (loc_A signifie que le produit est passé par la localisation A). Il est possible de construire des données de comportement variées : intervalles de valeurs, ensembles, séquences augmentées (avec des types, des valeurs numériques,...). Notre but sera d'utiliser ces données de comportement comme support pour labelliser chaque trajectoire contextualisée avec une classe anormale et normale (négative/positive). Cette classification permettra ensuite la découverte de motifs caractéristiques de la base négative.

Le langage des descriptions des traces, obtenues avec la fonction $description(t)$, ainsi que son codage est à définir en fonction des informations que l'on veut utiliser pour définir les anomalies de comportements. De nombreuses possibilités sont offertes via des langages de motifs très bien étudiés et associés à des contraintes bien définies : itemsets (fréquents, clos, maximaux, rares), séquences, graphes, etc. L'ensemble des descriptions des traces est appelé un *contexte de fouille* et sert de support à la fouille de motifs. La troisième colonne de la Table 3.1 montre une exemple de descriptions sous la forme d'ensembles. Par exemple, la première trajectoire contextualisée possède la description $\{vente, nombre\_stockage\_2, passage\_loc\_A, passage\_loc\_B, passage\_loc\_C\}$, qui signifie que le produit a été vendu, stocké deux fois, puis est passé par A, B et C. Un motif fréquent est un ensemble qui apparait souvent dans ces descriptions. Par exemple,

$\{nombre\_stockage\_2, passage\_loc\_A, passage\_loc\_B\}$ est présent dans les trois trajectoires, il a un support de 3.

Nous proposons un cas de codage de propriétés en transformant les traces unitaires dans un contexte de fouille d'itemsets. Ces propriétés intègrent des connaissances de haut niveau sur les objets qui peuvent être de nature spatio-temporelle ou symbolique. A partir de ces traces unitaires brutes, nous pouvons construire des contextes de fouille via une opération de transformation telle que $\mathcal{T} \rightarrow D$ qui vont constituer les descriptions. Le contexte de fouille D consiste en un ensemble d'instances (descriptions), comme présentées dans la troisième colonne de la table 3.1. Ces descriptions peuvent être de types variées et hétérogènes (numériques, symboliques, séquentiels, etc.) à partir du moment où ces descriptions peuvent être partiellement ordonnées et qu'il existe un algorithme (nommé solveur) pour les extraire. Nous utilisons dans nos exemples, le cas de bases de transactions, sous la forme de tables binaires où chaque transaction correspond à une trace unitaire et chaque propriété est une propriété booléenne.

Chaque propriété $p \in I$ est vue comme une fonction $p : \mathcal{T} \rightarrow \{true, false\}$. Une propriété consiste à évaluer une expression logique $\mathcal{C}$ (règles expertes, expressions régulières, présence d'évènements,etc.) construite à partir des traces unitaires et dont l'ensemble des propriétés ainsi générées forment la partie *description* d'une trajectoire, cette étape de génération est appelée étape d'échelonnage.

Soit une trace $t = \langle r_1, ..., r_i, ..., r_k \rangle$ composé d'une séquence de n-uplets $r_i = (a_1, ..., a_n)$ avec $a_i \in dom(A_i)$, un processus d'échelonnage va générer des propriétés de plusieurs manières que nous formalisons.

Definition 32 (Transformation par valeur) *Soit un attribut A_i et son domaine de valeur $dom(A_i)$, il s'agit de générer une propriété p_i par valeur $a_i \in dom(A_i)$ avec $p_i = (A_i, =, a_i), \forall a_i \in dom(A_i)$.*

Par exemple, si on prends l'attribut biz_loc de la table de traces 1.4 de l'introduction, une transformation par valeur génèrera : $p_1 = loc\_A$, $p_2 = loc\_B$ et $p_3 = loc\_C$.

Definition 33 (Transformation unique avec une règle) *Soit une règle experte x définie par une conjonction de prédicats $x = x_1 \wedge ... \wedge x_n$ où $x_i = vrai$ si le i-ème prédicat est vérifié. Un prédicat $x_i = (A_j, op, a_i)$ est une sélection sur un attribut A_j via un opérateur $op \in \{=, <, >\}$ et $a_j \in dom(A_j)$ d'une trace t. Si une trace t vérifie une règle x, c'est-à-dire que l'ensemble des prédicats $x_i = vrai$, alors $x(t) = vrai$.*

Il s'agit ici d'exprimer une propriété sous la forme d'une règle comme par exemple $p_4 = ('biz\_step', '=', 'STORING') \wedge ('biz\_loc', '=', 'loc\_B')$ qui est vraie si le produit a été stocké à la localisation B.

Definition 34 (Transformation unique par agrégat) *Pour un attribut donné, on peut générer un unique attribut selon un agrégat. La propriété est vraie si l'agrégat possède une valeur donnée. Cela correspond à un prédicat $x_i = (A_i, op, ag(dom(A_i)), value)$ où A_i est l'attribut, op un opérateur, $ag = \{count, avg, median, sum\}$ une fonction d'agrégat sur*

le domaine de valeur de A_i et value une valeur numérique qui sert de comparaison par rapport à $ag(dom(A_i))$ selon l'opérateur op et retourne vrai ou faux. Dans le cas d'un attribut catégoriel, count compte le nombre d'occurrences d'un attribut.

Par exemple, pour connaître le nombre de sites que la trace unitaire a parcourue, on peut définir cette propriété $p_5 = (\prime biz\_loc\prime, \prime > \prime, \prime count\prime, \prime 2\prime)$ qui est vraie si le produit est passé par strictement plus de 2 sites.

Definition 35 (Transformation inter-ordinale) *Soit un attribut numérique A_i et son domaine de valeurs $dom(A_i)$, la transformation interordinale consiste a générer l'ensemble de propriétés $P = \{(a_1, \leq), (a_1, \geq), ..., (a_n, \leq), (a_n, \geq)\}$ où $a_i \in dom(A_i)$ et n le nombre de valeurs dans $dom(A_i)$. Pour un attribut A_i ayant n valeurs, le nombre de propriétés générées sera donc de $2 \times n$.*

Pour illustrer cette transformation, nous assignons un prix à chaque produit.

epc_id	prix (p)	$p \leq 20$	$p \leq 30$	$p \leq 40$	$p \leq 100$	$p \leq 150$	$p \geq 20$	$p \geq 30$	$p \geq 40$	$p \geq 100$	$p \geq 150$
1	20	×	×	×	×	×	×				
2	20	×	×	×	×	×	×				
3	30		×	×	×	×	×	×			
4	40			×	×	×	×	×	×		
6	100				×	×	×	×	×	×	
7	100				×	×	×	×	×	×	
8	150					×	×	×	×	×	×
9	150					×	×	×	×	×	×

FIGURE 3.3 – Contexte de fouille généré à partir d'un échelonnage inter-ordinal

Si on extrait les motifs clos de la Table 3.3, on obtient des intervalles de valeurs par exemple le motif $\{prix \geq 20, prix \leq 150\}$ décrit toutes les traces et correspond à l'intervalle de valeur $[20, 150]$. Le motif $\{prix \geq 40, prix \leq 100\}$ quand à lui correspond aux traces $4, 6, 7$ dont le prix est dans l'intervalle $[40, 100]$.

Definition 36 (Transformation par discrétisation) *Soit un attribut A_i de la trace t_j noté $t_j(A_i)$ et son domaine de valeur $dom(t_j(A_i))$, on note $\Omega(A_i) = \bigcup_{j=1}^{k} dom(t_j(A_i)), k = |\mathcal{T}|, t_j \in \mathcal{T}$ le domaine de valeurs possibles de l'attribut A_i dans toutes les traces. Le processus de discrétisation crée n partitions $\{pa_1, ..., pa_n\}$ avec pa_i une propriété qui est vraie si l'attribut d'une trace possède des valeurs incluses dans pa_i.*

Le processus de discrétisation peut être très simple (diviser en n intervalles de taille $(max - min)/n$), basé sur des mesures statistiques comme $\mathcal{X}^2$ [52] ou sur l'entropie [27]. Un exemple est donné dans la Table 3.4.

Nous avons présenté un ensemble de méthodes pour la génération de propriétés booléennes à partir des attributs des traces. Il est également possible de procéder à des traitements faisant intervenir des sources de données externes. On peut se servir de modèles, comme le modèle de filière ou des modèles mathématiques. Une transformation par modèle génère, à partir d'un modèle M, une ou plusieurs propriétés. Comme un ensemble de règles peut être vu comme un modèle, nous proposons de définir d'autres types de modèles :

epc_id	prix (p)	$p \in [20,46]$	$p \in]46,72]$	$p \in]72,98]$	$p \in]98,124]$	$p \in]124,150]$
1	20	×				
2	20	×				
3	30	×				
4	40	×				
6	100				×	
7	100				×	
8	150					×
9	150					×

FIGURE 3.4 – Contexte de fouille $\mathcal{D}$ après une discrétisation

Definition 37 (Transformation par modèles) *Soit une trace t, un ensemble de propriétés P est généré de la manière suivante : $M(t) = \{p_1, ..., p_2\}$. Les modèles peuvent être varié :*

- *Clusters : Soit un ensemble de points $(z_1, ..., z_n)$ issus de n traces, calculés à partir d'attributs de chaque trace, un algorithme de clustering partitionne les n points en k partitions $S = \{s_1, ..., s_k\}$ telle qu'une mesure de distance entre les points de chaque partition est minimale. Une propriété p associée à la trace t est générée pour chaque partition s_i où le point correspondant z_i est inclus, soit $\forall s_i \supseteq z_i$.*
- *Classifieurs : A partir du même ensemble de points $z = (z_1, ..., z_n)$ calculés à partir des traces, il est possible de construire un classifieur M (avec un SVM par exemple) si chaque trace possède une classe c. Dans le cas où l'on a deux classes $c = \{+, -\}$, et un nouveau point z_1, on a donc un modèle $M(z_1) \rightarrow \{+, -\}$, la propriété générée est donc la classe obtenue en sortie.*
- *Modèle de filière : Le modèle de filière que l'on a défini précédemment peut permettre de générer des propriétés booléennes. Par exemple, si les transitions possèdent l'information du temps de livraison moyen entre deux sites, alors on pourra générer une propriété pour les traces unitaires qui possèdent ces deux sites.*

Le choix du type de transformation est primordiale pour pouvoir obtenir des motifs intéressants. Ce type de transformation peut être sélectionné manuellement, comme nous le proposons dans notre prototype, pour construire des contextes de fouille et éventuellement les faire évoluer selon les résultats. Nous montrons notamment un cas d'application dans les jeux vidéo. Nous avons conçu quelques propriétés pour découvrir des erreurs de stratégies grâce à l'expertise de joueurs confirmés. Un exemple simple est exposé dans la Figure 3.5. Tout d'abord, cet exemple nous montre trois propriétés qui représentent chacune un intervalle de valeur du nombre de morts d'un joueur durant la partie qui est un aspect du jeu déterminant pour la victoire (plus un joueur meure, moins il a d'expérience et donc de capacité pour se développer durant le jeu). Deux propriétés indiquent si un objet A (ou B) a été acheté. On voit dans cet exemple simple que les traces ayant la classe positive appartiennent à un joueur étant mort peu de fois (de 0 à 5) et ayant acheté l'objet A, tandis que les traces ayant la classe négative appartiennent à un joueur étant mort souvent (de 6 à 10 et de 10 à 15) et ayant acheté l'objet B. On peut en déduire alors qu'acheter l'objet B est une mauvaise stratégie.

Toutes ces techniques de transformation des données permettent d'exprimer de nom-

tid	$morts \in [0,5]$	$morts \in [6,10]$	$morts \in [10,15]$	$objet\_A\_acheté$	$objet\_B\_acheté$	classe
1	×			×		+
2	×			×	×	+
3	×			×		+
4		×			×	-
5		×			×	-
6			×		×	-

FIGURE 3.5 – Contexte de fouille $\mathcal{D}$ dans le domaine du jeu vidéo

breux scénarios. Dans la sous-section suivante nous présentons les possibilités d'extraction de connaissance à partir des contextes de fouille.

3.3.3 Extraction de connaissances à partir des contextes de fouille

Les tables binaires générées par les processus de transformation peuvent être analysées par les algorithmes de fouille de motifs ensemblistes, présentés dans l'état de l'art. L'ensemble des solveurs présentés dans l'état de l'art portant sur des données transactionnelles peuvent être utilisés pour l'extraction de motifs. Dans notre cas précis, il faut souligner que l'on peut se restreindre à l'extraction de motifs fermés qui sont une version condensée qui permet de réduire le nombre de résultats tout en conservant l'information. Dans nos expérimentations, nous utiliserons l'algorithme CHARM proposé par [96] car nous encodons les traces avec des propriétés booléennes. Comme nous l'avons vu, il est possible de construire des contextes de fouille ayant un autre langage de motifs (comme les motifs séquentiels par exemple) et ainsi appliquer d'autres solveurs adaptés. Par exemple, une généralisation directe a été proposée par Ganter et Kuznetsov [31] pour traiter des données hétérogènes dont les descriptions possibles peuvent être ordonnées au sein d'un demi-treillis. Cette généralisation est suffisante pour traiter les données de type numérique, séquentiels, ordres partiels et graphes dans certaines conditions.

Nous exposons l'ensemble des définitions qui nous permettent d'obtenir à partir d'une base de descriptions D un ensemble de motifs X. Le cadre de la fouille de données, établi par Mannila et al. [65] permet de décrire une tâche de fouille de données en la ramenant au calcul de la théorie $Th(\mathcal{D},\mathcal{L},\mathcal{C})$.

Definition 38 (Théorie)

$$Th(D,\mathcal{L},\mathcal{C}) = \{X \in \mathcal{L} | \mathcal{C}(X,D) = true\}$$

où D est la base de descriptions, $\mathcal{L}$ le langage de motifs, et $\mathcal{C}$ est une conjonction de contraintes qui doit être vérifiée par tous les motifs X extraits.

Nous revenons sur le cas précis des ensembles de propriétés. Un ensemble $X \subseteq P$ est appelé un itemset. Le support d'un itemset X dans $\mathcal{D}$ noté $supp(X)$ correspond au nombre de descriptions des traces qui contiennent X soit $supp(X) = |\{X | X' \in \mathcal{D}, X \subseteq X'\}|$. Un itemset fermé est tel qu'il n'existe pas d'itemsets de même support qui le contiennent. On appelle un motif fermé fréquent un itemset fermé ayant un support supérieur à un seuil min_sup. Les ensembles fréquents permettent d'extraire des co-occurrences de propriétés fréquemment observés dans les traces. Selon le codage des

propriétés que l'on aura fait via les processus d'échelonnage, la sémantique des ensembles extraits sera différente.

A ce stade, nous avons la possibilité de sélectionner, transformer puis analyser les traces unitaires de produits avec des algorithmes de fouille de données bien étudiés appliqués sur la description de chaque trace. Nous avons défini, pour chaque trajectoire, une séquence de sites qui permet de décrire le comportement du produit. Cela nous permet de servir de support au calcul de l'anormalité d'une trajectoire. Pour cela, nous devons avoir une structure de données qui va modéliser ce qui est normal. Cette structure, que nous appelons modèle de filière, prend la forme d'un graphe orienté, et est décrite dans le chapitre suivant.

3.3.4 Filtrage, interprétation et visualisation des motifs

Après que les traces unitaires soient transformées en contexte de fouille et/ou que l'on ait extrait des motifs, il est possible d'imaginer une étape de filtrage. Une collection de traces $\mathcal{T}_m$ peut être construite à partir de $\mathcal{T}$ et d'opérateurs de sélection basés sur les données contenues dans les trajectoires ou les descriptions des traces et non directement sur les attributs de ces traces. Soit une base de traces $\mathcal{T}$, un opérateur de sélection basé sur les trajectoires et/ou les descriptions, donnera l'ensemble de traces $\mathcal{T}_m = \{t | \mathcal{C}_1(trajectory(t)) = true \vee \mathcal{C}_2(description(t)) = true, t \in \mathcal{T}\}$. Il est aussi possible de sélectionner une collection de traces simplement en récupérant les traces unitaires à partir des motifs. En effet, ceux-ci portent sur des descriptions qui sont assignées à des produits (et donc des traces). Il est également envisageable de combiner ces méthodes de sélection (construire une collection de traces unitaires à partir d'un ensemble de motifs, de motifs ayant certaines propriétés,etc).

Pour interpréter et visualiser les motifs, nous avons conçu une interface graphique sous la forme d'une plateforme web. L'analyste peut donc construire des contextes de fouille à partir d'une sélection de traces unitaires.

La partie de gauche de la Figure 3.6 permet de sélectionner le type de propriété à générer et de rentrer divers paramètres pour guider leur génération. Une fois validé, le contexte de fouille est généré et visualisé sous la forme d'une table. Un V vert indique que la propriété est vraie pour la trace de la ligne correspondante, et la croix rouge indique que la trace ne vérifie pas la propriété. En bas à gauche, deux options permettent à l'analyste soit d'extraire les motifs fréquents dans ce contexte de fouille, en spécifiant le seuil de fréquence minimum. Le résultat est affiché dans l'interface de la Figure 3.7.

A droite, on voit la liste des motifs fréquents extraits avec le support (le nombre de traces qui sont décrites par le motif) et la description du motif (dans cet exemple, le motif contient les ensembles de sites ainsi que les jours durant lesquels les produits concernés ont voyagé). A gauche, le modèle de filière est affiché sur une carte du monde, avec comme noeuds les sites (qui possèdent une couleur différente par type de sites) et comme arêtes les transitions possibles entre les sites. Une fois que l'on clique sur le motif, les séquences de sites qui sont assignées à chaque trace (le résultat de la fonction $trajectory(t)$) sont récupérées et sont affichées sous forme d'arêtes en gras.

La Figure 3.8 montre le résultat de notre méthode du prochain chapitre, avec la

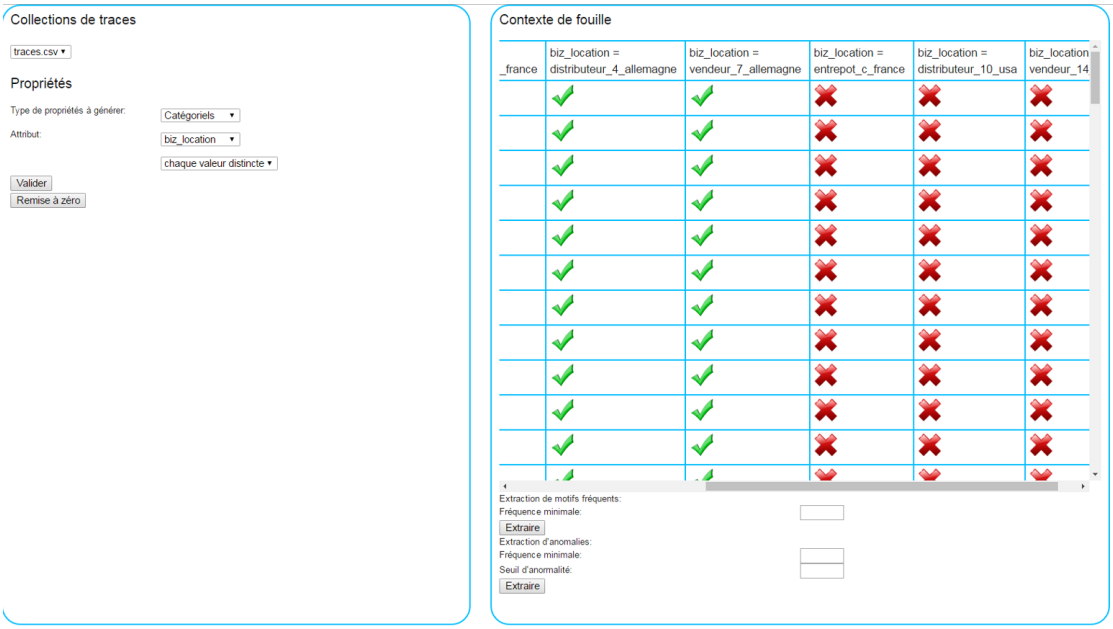


FIGURE 3.6 – Interface de création de contextes de fouille

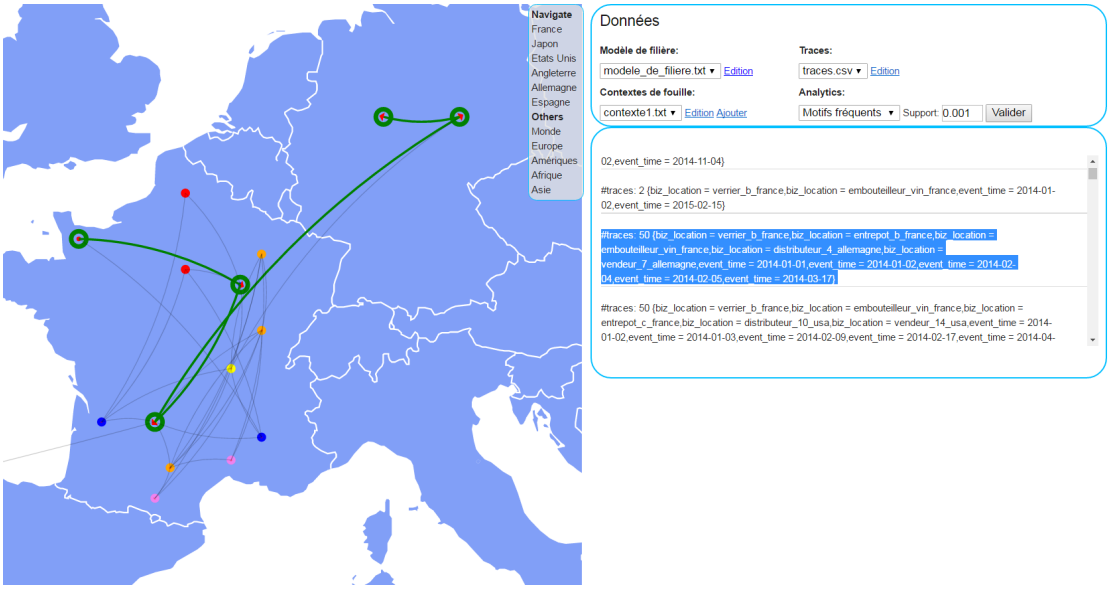


FIGURE 3.7 – Interface de visualisation des motifs fréquents

visualisation d'un motif anormal, avec son score d'anormalité, son support et la description. Quand on clique dessus, on obtient l'affichage de l'ensemble des séquences de sites décrites par ce motif. On observe que deux arêtes sont en rouge, c'est la section de la séquence de site problématique qui peut décrire un détournement de produit.

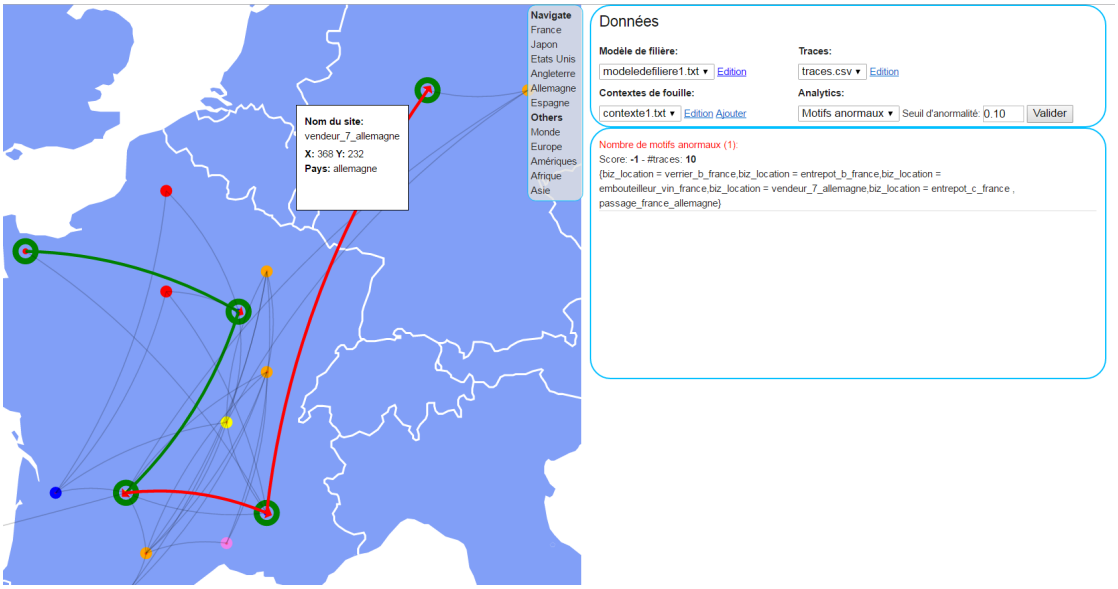


FIGURE 3.8 – Interface de visualisation d"un motif anormal

3.4 Conclusion

Durant l'état de l'art nous avons vu qu'un ensemble de méthodes permettait de stocker, modéliser, compresser et interroger les traces unitaires mais, à notre connaissance, aucune ne proposait de formaliser un cadre global d'analyse de traces unitaires en utilisant le processus d'extraction de connaissances à partir des données. Dans ce chapitre, nous avons tout d'abord présenté les données de traçabilité sous plusieurs aspects : le contexte manufacturier et son vocabulaire spécifique, ainsi que les données synthétiques et réelles que nous allons utiliser pour nos expérimentations. Nous avons ensuite décrit comment le processus d'extraction de connaissances peut être adapté à la fouille de traces unitaires. Plus particulièrement, nous avons détaillé un ensemble de méthodes de transformation des traces unitaires. Celles-ci permettent de construire des contextes adaptés à la fouille de données, sous forme de tables binaires, intégrant de la connaissance experte. Ces contextes de fouille traduisent des scénarios variés, suivant la sémantique des propriétés booléennes, et servent de supports pour l'application de méthodes de fouille de motifs ensemblistes, comme l'extraction d'évènements co-occurrents fréquents. Une fois, les motifs extraits, nous voyons qu'il est possible d'utiliser ces motifs pour interroger les autres structures de données, notamment en récupérant les ensembles de traces unitaires décrites par le motif, ou encore déterminer dans le modèle de filière les chemins parcourus par les produits correspondants aux traces. Au sein de ce cadre méthodologique, il est possible d'insérer de nouvelles méthodes de fouille de données, utilisant les différents concepts. Dans le chapitre suivant, nous proposons, via notre deuxième contribution, d'utiliser le modèle de filière (les données captables) conjointement aux données captées pour extraire des évènements co-occurrents fréquents particuliers. Ceux-ci permettent de

décrire les collections de traces négatives, c'est-à-dire dont le comportement des produits est jugé anormal par rapport au modèle de filière. Nous montrons également comment le modèle de filière, que nous formalisons en détail, peut être construit et mise à jour de manière automatique ou par les analystes.

Chapitre 4

Découverte de motifs intelligibles et caractéristiques d'anomalies dans les traces unitaires

Nous avons présenté dans le chapitre précédent un ensemble de notions centrales : les traces unitaires, correspondant aux données captées, les trajectoires contextualisées, correspondantes à chaque trace unitaire transformée, les motifs, permettant de décrire des collections de traces partageant des descriptions communes et le modèle de filière, décrivant les données captables. Les données captables peuvent ne pas correspondre aux données captées, en effet le détournement d'un ensemble de produits entrainera des séquences d'évènements non prévues (par exemple si des produits disparaissent à partir d'un site pour réapparaitre sur un autre site non prévu par le modèle de filière). Nous souhaitons alors mesurer la déviance d'une trace unitaire par rapport aux comportements normaux, soit la proportion de transitions non prévues entre les sites d'une trace. Nous utilisons le modèle de filière pour partitionner la base de traces unitaires en deux : les traces normales, décrivant des comportements jugés conformes au regard du modèle de filière, et les traces anormales, correspondant aux comportements jugés déviants. Nous souhaitons extraire les contextes qui expliquent chaque type de comportement, et plus particulièrement ceux anormaux. Pour cela, nous utilisons la fouille de données pour découvrir des motifs discriminants de traces négatives, susceptibles donc de décrire des anomalies. Nous exposerons les résultats de notre méthode sur des jeux de données synthétiques et réels. D'une part, nous utilisons un jeu vidéo de gestion de réseaux manufacturiers que nous utilisons pour générer des ensembles de traces réalistes qui serviront à l'analyse du passage à l'échelle de notre méthode selon plusieurs paramètres (nombre d'items, nombre d'objets, taille du modèle de filière et des séquences de sites). Nous avons également conçu trois scénarios différents avec des anomalies insérées pour démontrer que notre méthode les extrait. D'autre part, nous utilisons des traces comportementales réelles d'un autre jeu vidéo pour découvrir des motifs caractéristiques de stratégies erronées.

4.1 Exprimer de la connaissance du domaine via un modèle de filière

Le modèle de filière est la structure de données que nous utilisons pour décrire la connaissance que possèdent les analystes du domaine manufacturier. Nous proposons dans ce manuscrit d'utiliser un modèle de filière globale décrivant toute la chaîne logistique, mais il est possible d'imaginer que les analystes de chaque acteur logistique construisent des visions spécifiques. Il désigne les comportements attendus que devront effectuer les produits dans le système manufacturier et qui sont captables. La construction de ce modèle pose plusieurs problématiques importantes : Est ce que ce modèle doit être généré manuellement, automatiquement ou les deux ? Si on le génère automatiquement, à partir de quelles sources de données doit on générer ce modèle ? Quelles sont les vues partielles du modèle de filière intéressantes pour les utilisateurs ? Doit-on générer l'évolution de ce modèle de filière suivant les résultats d'analyses ? Un champ de fouille de données s'intéresse particulièrement à la construction de modèles à partir de traces d'exécution, la fouille de workflow [90]. Ces méthodes peuvent être intéressantes dans le contexte manufacturier pour décrire les processus appliqués aux produits. Dans notre cas, nous nous tenons à une description d'un modèle simple pour la définition de flux d'objets. Il convient de décrire les différentes manières de construire ce modèle en premier lieu, puis de décrire les différentes possibilités d'analyse offertes par le modèle en lui même (via l'application de méthodes de fouille directement sur ce modèle) ou par ses relations avec les autres données (traces, trajectoires ou motifs). Nous explorons les possibilités d'évolution de ce modèle de filière, et l'intérêt de conserver un ensemble de modèles de filière pour l'analyse des systèmes manufacturiers.

4.1.1 Construire un modèle de filière

Nous avons vu que l'analyse du système manufacturier suppose de faire des hypothèses sur la structure de ce système. Étant donné que le système manufacturier comporte différentes sites et que les produits parcourent ces sites, la structure la plus classique pour modéliser ces informations est un graphe orienté.

Definition 39 (Modèle de filière) *Chaque enregistrement est réalisé sur un site, qui est un attribut d'enregistrement r particulier (noté $site(r)$) des traces. L'ensemble des sites forme un graphe $G = (V, E)$ avec V les nœuds et $E \subseteq V \times V$ les arêtes, que l'on appelle modèle de filière.*

Un enregistrement r correspond à une action effectuée à un instant t et à un site $v \in V$. Ce modèle peut être définit manuellement, grâce à une interface graphique par exemple, en construisant les sites et les arêtes. Cependant, dans certains cas les transitions entre les noeuds peuvent être construites automatiquement, en s'appuyant sur une collection spécifique de traces unitaires que l'on sélectionne via une requête.

Soit un sous-ensemble de traces $\mathcal{T}_m \subseteq \mathcal{T}$ issus d'une requête de sélection quelconque. Par exemple, on peut sélectionner l'ensemble des traces d'un constructeur donné de produits fabriqués durant un trimestre et ayant des caractéristiques physiques bien précises

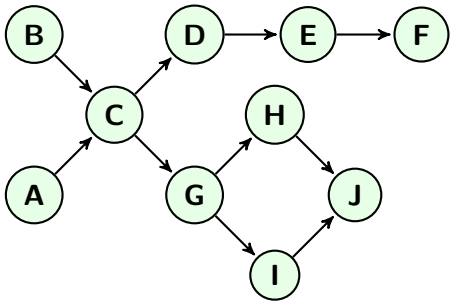


FIGURE 4.1 – Exemple d'un modèle de filière comportant 10 sites

(taille, forme, poids). Les noeuds de notre modèle de filière étant les positions traversées par les produits, on génère alors un noeud distinct par élément de la trajectoire, $\forall v_i \in trajectory(t)$, on ajoute v_i à l'ensemble V.

Definition 40 *Soit un modèle de filière $G = (V, E)$, avec V les noeuds et E les arêtes, l'ensemble des noeuds est égal à :*

$$V = \bigcup_{i=0}^{i=|\mathcal{T}|} trajectory(t_i), t_i \in \mathcal{T}_m$$

Nous procédons ensuite à la construction des arêtes E qui symbolisent les transitions entre deux noeuds. Les trajectoires de $\mathcal{T}_m$ possèdent les informations de passages entre deux noeuds ce qui peut servir de support pour la création des arêtes dans G. Il s'agit de pondérer chaque paire de noeuds par son nombre d'occurrences dans l'ensemble des séquences de sites de $\mathcal{T}^m$. Ainsi, le poids assigné aux deux sites indiquera le nombre total de passages entre ces sites. Pour supprimer les arêtes non fréquentes, nous calculons la moyenne total des poids des arêtes puis nous supprimons l'ensemble des arêtes dont le poids est inférieur à la moyenne. Il peut être possible d'envisager d'autres méthodes de suppression des arêtes par la suite pour concevoir des modèles avec des sémantiques différentes. Dans le premier cas, nous avons construit manuellement un modèle de filière simple pour les expérimentations sur des données synthétiques. Nous l'utilisons également pour simuler des détournements de produits, en vue de voir si notre méthode les retrouve efficacement. Dans le deuxième cas, nous utilisons des données réelles de traces de jeu où l'ensemble des arêtes est construit suivant le processus précédent.

4.2 Extraire des motifs caractéristiques d'anomalies

Une fois que le processus de transformation des traces unitaires $\mathcal{T}$ en contexte de fouille $\mathcal{D}$ est effectué, il est possible d'appliquer un ensemble de solveurs analysant les contextes booléens, comme la fouille d'itemsets fréquents, clos ou maximaux. Dans cette section, nous proposons une méthode pour extraire des descriptions intéressantes de collections de traces unitaires. Ces descriptions sont issues des trajectoires

contextualisées décrites précédemment comme des paires $(trajectory(t), description(t))$ par exemple la première trajectoire contextualisée de la Table 4.2 est $(\langle A, C, D, E, F\rangle, \{vente\_france, distributeur\_D\})$. Dans notre cas, une description est intéressante si elle apparaît dans de nombreuses traces anormales, autrement dit que ne respectent pas le modèle, et très peu dans les classes normales, connu dans la littérature sous le nom de motif émergent [25]. Il est possible d'imaginer d'autres langages de descriptions comme les séquences, les intervalles numériques ou les graphes.

Pour définir la mesure d'anomalie d'un motif, nous utilisons l'ensemble des trajectoires de chaque trace contextualisée, ainsi qu'une matrice M construite à partir d'un modèle de filière G. Cette matrice est de taille $|V| \times |V|$ et chaque cellule $M(v_1, v_2)$ possède une valeur comprise entre 0 et 1 où 0 indique que la transition entre le site v_1 et le site v_2 est normale et 1 que la transition est anormale. Les notions de normalité peuvent être diverses, par exemple un cas simple permet de dire que s'il n'y a pas de transition prévue entre les deux sites par le modèle de filière alors la transition est anormale (donc égale à 1) sinon la transition est normale (égale à 0). Une fois cette matrice générée, elle sert de support au calcul d'un score d'anomalie pour une trace unitaire :

Definition 41 (Score d'anomalie pour une trace unitaire) *Soit une trace t, et M une matrice pondérée,*

$$\mu(t, M) = \frac{\sum_{i=0}^{i=|trajectory(t)|-1} M(t_i, t_{i+1})}{|trajectory(t)| - 1} \tag{4.1}$$

$|trajectory(t)|$ est le nombre de sites de la trajectoire de t (sa taille).

Ce score calcule la moyenne des scores de chaque transition issus de la matrice M pour une trace t. Si toutes les transitions sont anormales, alors le score d'anomalie pour une trace t sera maximal, soit 1. C'est le cas de la 5ème trajectoire contextualisée présentée dans le Tableau 4.2. On a $trajectory(t_5) = \langle B, D, H, I, F\rangle$ avec les transitions (B, D),(D, H),(H, I) et (I, H) qui ne sont pas présentes dans le modèle de filière. Notre objectif est de construire une partition de notre base initiale de traces unitaires. Pour cela, à l'aide d'un seuil θ compris entre 0 et 1, nous produisons deux ensembles de traces unitaires, un ensemble de traces positives et un ensemble de traces négatives, en fonction du score d'anomalies de chaque trace. Ainsi, plus le score est élevé, plus la base de traces négatives sera petite et contiendra des traces fortement anormales. Nous les construisons de la façon suivante :

Definition 42 *(Traces positives et négatives) Soit t une trace unitaire, M une matrice pondérée, un seuil θ et un score d'anomalie $\mu(t, M)$, on a la partition de $\mathcal{T}$ suivante :*

$$\mathcal{T}^+ = \{t | t \in \mathcal{T}, \mu(t, M) \leq \theta\}$$
$$\mathcal{T}^- = \{t | t \in \mathcal{T}, \mu(t, M) > \theta\}$$

Notre objectif est de décrire l'ensemble des traces négatives c'est-à-dire d'extraire les ensembles de propriétés présents dans les descriptions de chaque trace qui décrivent de

pid	Trajectoire	Description	Score trans.	μ
1	$\langle A,C,D,E,F\rangle$	$\{vente\_france, distributeur\_D\}$	0	0
2	$\langle A,C,D,F\rangle$	$\{vente\_france, distributeur\_D\}$	0.33	0.0825
3	$\langle B,C,G,J\rangle$	$\{vente\_usa, distributeur\_G\}$	0.33	0.0825
4	$\langle B,C,H,F\rangle$	$\{vente\_france, distributeur\_H\}$	0.66	0.41
5	$\langle B,D,H,I,F\rangle$	$\{vente\_france, distributeur\_H\}$	1	0.81

FIGURE 4.2 – Cinq trajectoires contextualisées avec les scores d'anomalie respectifs

nombreuses traces négatives mais peu de traces positives. Pour cela, nous construisons simplement les bases de descriptions $\mathcal{D}^+$ et $\mathcal{D}^-$ issues des bases de traces positives et négatives :

Definition 43 *(Descriptions positives et négatives)*

$$\mathcal{D}^+ = \{description(t) | t \in \mathcal{T}^+\}$$
$$\mathcal{D}^- = \{description(t) | t \in \mathcal{T}^-\}$$

Avec $\theta = 0.60$, on a une base $\mathcal{D}^-$ avec deux descriptions semblables $\{vente\_france, distributeur\_H\}$. Nous voyons que ces deux ensembles de propriétés sont présents uniquement dans cette base, et jamais dans le reste des descriptions. C'est ce genre d'ensemble que nous recherchons. Ce partitionnement permet de se baser sur des méthodes de découverte de motifs émergents [25]. Nous nous basons alors sur cette mesure d'émergence qui est d'autant plus forte que le motif apparait dans une classe choisie, cependant nous utilisons dans un premier temps une mesure normalisée entre $[-1,+1]$.

Definition 44 (Mesure d'émergence normalisée) *Soit un motif X, la base $\mathcal{T}^+$ et la base $\mathcal{T}^-$, on a la mesure d'émergence normalisée suivante :*

$$\phi(X) = \frac{|supp_{\mathcal{D}^+}(X)| - |supp_{\mathcal{D}^-}(X)|}{|supp_{\mathcal{D}^+}(X)| + |supp_{\mathcal{D}^-}(X)|}$$

Si la valeur d'émergence du motif est strictement inférieure à 0, alors on a un motif émergent de description d'anomalie. Si la mesure vaut 0, alors il y a autant de traces dans $\mathcal{T}^-$ que dans $\mathcal{T}^+$ qui respectent ce motif. Nous pouvons utiliser d'autres mesures, comme le $\mathcal{X}^2$ [37] présentée dans l'état de l'art, que nous avons utilisé dans nos expérimentations.

Finalement, nous exprimons notre problématique à partir de l'ensemble des définitions décrit précédemment comme suit :

Problème 2 (Découverte de motifs de description d'anomalie) *Soit une base de traces unitaires $\mathcal{T}$ et un contexte de fouille D produit à partir de propriétés P, un modèle de filière $G = (E,V)$, un seuil d'anormalité θ, et un support minimum min_sup, nous cherchons l'ensemble de tous les motifs émergents de description d'anomalie X de D tel que $supp(X) > min\_sup$ et $\phi(X) < 0$.*

Algorithm 1 DESCANOM Pseudo code

Require: Un modèle de filière G, θ : seuil d'anormalité, min_sup support minimal, $\mathcal{T}$ base de traces unitaires.

Ensure: S motifs caractéristiques de descriptions d'anomalies

1: $M \leftarrow ComputeOutlierMatrix(G)$
2: $\mathcal{A} \leftarrow ComputeTrajectories(\mathcal{T})$
3: $\mathcal{D} \leftarrow ComputeDescriptions(\mathcal{T})$
4: $\mathcal{T}^+ \leftarrow \{t | t \in \mathcal{T}, \mu(a, M) \leq \theta\}$
5: $\mathcal{T}^- \leftarrow \{t | t \in \mathcal{T}, \mu(a, M) > \theta\}$
6: $\mathcal{D}^+ \leftarrow \{description(t) \in \mathcal{D} | t \in \mathcal{T}^+\}$
7: $\mathcal{D}^- \leftarrow \{description(t) \in \mathcal{D} | t \in \mathcal{T}^-\}$
8: $MC \leftarrow CHARM(\mathcal{D}, min\_sup)$
9: **for all** $X \in MC$ **do**
10: $S \leftarrow S \cup X$ si $\phi(X) < 0$.
11: **return** S

Le pseudo code de notre méthode est présenté par l'Algorithme 1. Pour extraire les motifs émergents, nous utilisons le solveur CHARM [96] qui permet d'extraire l'ensemble des motifs fermés, c'est-à-dire des motifs non redondants où le maximum de traces unitaires partagent le maximum de propriétés communes. En effet, Plantevit et Crémilleux [76] ont démontré que les motifs fermés sont ceux qui maximisent les mesures basées sur le support, ce qui est le cas de la mesure d'émergence. L'implémentation de CHARM utilisé est celle présente dans la librairie JAVA SPMF [7] qui propose environ 120 algorithmes de fouille de motifs (fouille d'itemsets et de séquences principalement). Nous avons utilisé une machine HP Zbook 17, avec 16 go de RAM et un processeur Intel Core i7-4710MQ 2.5 GHz, 4 cœurs. Un autre solveur pourra être utilisé dans le cas où nous encodons les descriptions sous un autre domaine de motifs. Nous restons dans le cadre des itemsets, mais on notera qu'une généralisation directe a été proposée par Ganter et Kuznetsov [31] avec les structures de patrons. Elle sera utilisé dans le chapitre suivant. Arimura et Uno [7] proposent également un cadre général pour la découverte de motifs clos pour des sous-ensembles ordonnés selon l'inclusion ensembliste et possédant une certaine structure (séquences, graphes, images).

4.3 Définition d'une mesure d'anormalité

Nous pouvons abstraire la mesure d'anormalité précédente qui permet de partitionner notre base de traces en deux. En effet, cette mesure consiste à calculer la proportion de transitions non présentes dans le modèle de filière pour une trace à partir des données comportementales. Cela revient à compter dans la matrice d'adjacence M de dimension $|V| \times |V|$, du graphe G le nombre de 0 (pas de transitions) par rapport au nombre de transitions totales de la séquence. Nous pouvons alors proposer une généralisation en

7. http ://www.philippe-fournier-viger.com/spmf/

considérant que la valeur $M(i,j)$ de la matrice correspond à la valeur d'anormalité d'une transition $M(i,j) \in [0,1]$ où 0 la transition est normale, et 1 la transition est anormale. Nous pouvons alors définir notre score d'anomalie précédent avec la définition de la valeur d'une transition pour une séquence $a = \langle a_1, ..., a_n \rangle$.

Definition 45 *Valeur d'une transition*

$$M(v_i, v_j) = \begin{cases} 0 & si\ (v_i, v_j) \in G \\ 1 & sinon \end{cases}$$

L'intérêt de cette méthode est de proposer une mesure d'anormalité unique et de reporter la définition de la normalité d'une transition dans la construction de la matrice M. Nous proposons ainsi une mesure d'anormalité plus pertinente qui tenant compte de la distance entre deux sites. Étant donné que les produits sont transportés physiquement, plus ils réapparaissent loin dans le système de distribution (c'est-à-dire qu'il y a de nombreux sites à parcourir avant de relier le site du départ jusqu'au site où le produit réapparaît), plus c'est anormal. S'il n'est pas possible de relier les noeuds par un chemin prévu, alors l'anomalie est majeure. En effet, si deux sites ne sont pas reliés par une arête, nous savons qu'il y a un problème d'enregistrement. Or, si l'on a $\langle A,G \rangle$ on peut en déduire que l'anomalie se situe au niveau du site C uniquement. Cependant, si l'on a $\langle A, J \rangle$ le nombre de possibilités augmentent puisqu'il se peut que le produit soit passé par C,G,H puis J, ou par I. On voit donc que pour relier A à G le plus court chemin est de longueur deux alors que pour relier A à J, le plus court chemin est de 4. La valeur $M(v_i, v_{i+1})$ d'une trace sera donc d'autant plus forte que les noeuds peuvent être reliés par un plus court chemin qui est en fait le plus long chemin. Une transition entre deux noeuds aura un degré d'anormalité maximum quand ces deux noeuds n'ont aucun chemin entre eux possibles. Soit $path(v_1, v_2)$ le plus court chemin entre v_1 et v_2 avec $v_1, v_2 \in V$ où $path(v_1, v_2) = \infty$ s'il n'y a pas de chemin entre v_1 et v_2, avec $|path(v_1, v_2)|$ la longueur du chemin (nombre d'arêtes), et $max\_path(G)$ le chemin le plus long de G :

Definition 46 (Matrice d'anormalité avancée)

$$M(v_1, v_2) = \begin{cases} 1 & si\ path(a_1, a_2) = \infty \\ \frac{|path(v_1,v_2)-1|}{|max\_path(G)|} & sinon \end{cases}$$

Si $path(v_1, v_2)$ est égal à 1 alors $\frac{|path(v_1,v_2)-1|}{|max\_path(G)|}$ vaut 0 (les deux noeuds sont reliés directement par une arête) donc la transition est totalement normale. Plus $|path(v_1, v_2) - 1|$ se rapproche de la valeur du chemin maximal, plus les noeuds sont distant dans le graphe, et plus la transition est anormale. Le cas extrême est quand il n'y a pas d'arêtes, $M(v_1, v_2)$ vaut alors 1.

4.3.1 Mettre à jour un modèle de filière

Selon l'évolution du réseau manufacturier (construction de nouveaux sites ou établissement de nouveaux contrats), les experts peuvent facilement mettre à jour le modèle de

	A	B	C	D	E	F	G	H	I	J
A	1	1	0	0.25	0.50	0.75	0.25	0.50	0.50	0.75
B	1	1	0	0.25	0.50	0.75	0.25	0.50	0.50	0.75
C	1	1	1	0	0.25	0.50	0	0.25	0.25	0.75
D	1	1	1	1	0	0.25	1	1	1	1
E	1	1	1	1	1	0	1	1	1	1
F	1	1	1	1	1	1	1	1	1	1
G	1	1	1	1	1	1	1	0	0	0.25
H	1	1	1	1	1	1	1	1	1	0
I	1	1	1	1	1	1	1	1	1	0
J	1	1	1	1	1	1	1	1	1	1

FIGURE 4.3 – Matrice d'anormalité en fonction de la longueur des chemins

filière en ajoutant des noeuds ou des arêtes. Cependant, il se peut que lorsque les experts utilisent les algorithmes d'extraction de motifs anormaux, ils découvrent que le modèle de filière n'est en fait pas complet. Il est alors possible de proposer une méthode simple pour que les experts puissent corriger le modèle de filière lors de l'extraction de motifs, en choisissant les alertes levées erronés et corriger le modèle de filière automatiquement. L'idée est d'ajouter les arêtes manquantes au modèle de filière mais présentes dans les séquences du site des trajectoires contextualisées du motif.

Dans les différentes évolutions possibles pour la gestion du modèle de filière, on peut imaginer de le mettre à jour à partir des résultats de la fouille de motifs. Le modèle de filière, une fois généré, pourrait évoluer manuellement par le choix des analystes ou suivant les résultats de la fouille de données de traces. En effet, si les analystes constatent que le modèle est incomplet puisque le nombre d'anomalies est important ou qu'elles ne sont pas pertinentes, il sera alors intéressant de proposer un cadre formel d'évolution du modèle de filière. Un modèle de filière est mis à jour si un ou plusieurs noeuds ou arêtes sont ajoutés ou supprimés. Si le modèle de filière possède des attributs, la mise à jour peut porter sur les valeurs de ces attributs.

Soit un modèle de filière sous forme de graphe $G = (V, E)$, un modèle de filière mis à jour est un graphe $G' = (V', E')$. Les deux modèles de filière peuvent avoir des nœuds ou arêtes en commun ou être disjoints. Nous établissons un ensemble de définitions qui vont permettre de modéliser le processus de mise à jour d'un modèle de filière.

Definition 47 (Mise à jour d'un modèle de filière) *Soit deux modèles de filière G_1 et G_2, une fonction $maj(G_1) = G_2$ transforme G_1 en G_2 suivant une ou plusieurs transformations comme : ajout d'un ou plusieurs noeuds ou arêtes tel que $V_2 \subset V_1$ ou $E_2 \subset E_1$ et suppression d'un ou plusieurs noeuds tel que $V_1 \subset V_2$ ou $E_1 \subset E_2$.*

A partir du modèle de filière présenté précédemment nous pouvons dériver plusieurs modèles que l'on montre dans la Figure 4.4.

Premièrement le modèle de filière G_0 est mis à jour avec le retrait de deux noeuds B et F, soit $maj(G_0) = G_1 = (V_0 \setminus \{B, F\}, E_0 \setminus \{(E, F), (B, C)\})$. Dans un deuxième temps, nous avons une nouvelle mise à jour de G_1 à G_2 notée $maj(G_1) = G_2 = (V_1 \cup$

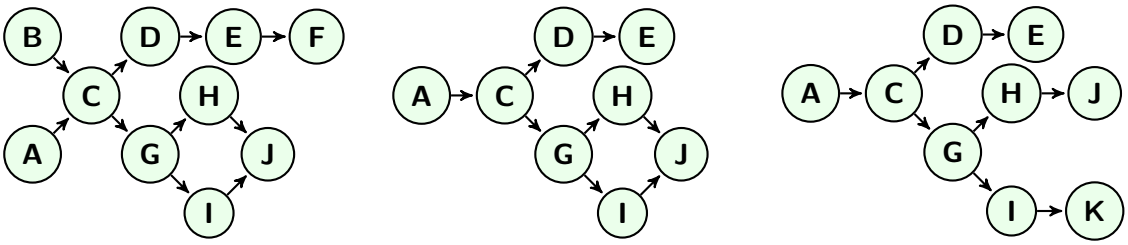


FIGURE 4.4 – Évolution d'un modèle de filière

$\{K\}, E_1 \setminus \{(I,J)\} \cup \{(I,K)\}$). Ces trois modèles de filière forment ce que nous appelons une séquence de modèles de filière. Elle décrit les mises à jour successives d'un modèle initial.

Definition 48 (Séquence de modèle de filière) *Soit $S = \langle G_0, G_1, ..., G_k \rangle$ une séquence de modèle de filière où G_0 est le modèle initial et G_k le dernier modèle. Il existe $k-1$ fonctions telles que $maj(G_i) = G_{i+1}$ qui indiquent les mises à jour d'un modèle de filière G_i pour obtenir le modèle G_{i+1}.*

Nous avons donc une séquence de modèle de filière telle que $S = \langle G_0, G_1, G_2 \rangle$ avec deux fonctions $maj(G_0)$ et $maj(G_1)$. Cette séquence de modèle de filière correspond à l'historique des modifications du modèle et permet aux experts d'effectuer des fouilles de contextes avec des modèles particuliers, ou de revenir en arrière, à des versions plus anciennes du modèle.

Nous proposons un cadre pour produire automatiquement une séquence de modèle de filière grâce aux résultats d'une fouille de contexte. On peut utiliser soit la fouille de motifs clos fréquents ou notre méthode de fouille de motifs discriminants. Chaque motif décrit un ensemble de trajectoires qui sont des séquences de sites. Cette séquence de sites, formalisée précédemment par $s = \langle s_1, ..., s_n \rangle$ où n est la taille de la séquence, s_i le i-ème site de s et (s_i, s_{i+1}) une transition entre s_i et s_{i+1}, peut correspondre à un ensemble de noeuds et d'arêtes par rapport à un modèle de filière G quelconque. Si $s_1 \in V_1$ et (s_1, s_2) avec $G_1 = (V_1, E_1)$ alors la séquence s a un site s_1 et une transition (s_1, s_2) présents dans le modèle de filière G_1. Soit $s(X)$ l'ensemble des séquences de sites de chaque trajectoire du motif X on peut formaliser deux opérations de mise à jour à partir d'un motif :

Definition 49 (Mise à jour d'un modèle de filière à partir d'un motif) *Soit un motif X, l'ensemble des séquences de sites $s(X)$, et un modèle de filière G, on définit l'ajout et la suppression d'éléments dans un modèle de filière G par :*

— *Ajout de nœuds :*

$$G = (V \cup \{s_i | s_i \in s, \forall s \in s(X), i = 1..n\}, E), \tag{4.2}$$

— *Suppression de nœuds (et impact sur les arêtes) :*

$$G = (V \setminus \{s_i | s_i \in s, \forall s \in s(X), i = 1..n\}, E), \tag{4.3}$$

— *Ajout d'arêtes :*

$$G = (V, E \cup \{(s_i, s_{i+1}) | (s_i, s_{i+1}) \in s, \forall s \in s(X), i = 1..n-1\}), \tag{4.4}$$

— *Suppression d'arêtes :*

$$G = (V, E \setminus \{(s_i, s_{i+1}) | (s_i, s_{i+1}) \in s, \forall s \in s(X), i = 1..n-1\}), \tag{4.5}$$

Par exemple, avec le motif $X_1 = \{X, Y\}$, $s(X_1) = \{< A, C, D, H >, < A, C, D, G >\}$, le modèle de filière G_0, et que l'on souhaite ajouter des arêtes, alors la fonction $maj(G_0)$ à partir du motif X_1 donne le graphe $G_4 = (V_4, E_4 \cup \{(D, H), (D, G)\})$ à gauche dans la Figure 4.5. Si on a un deuxième motif $X_2 = \{W, Z\}$ avec $s(X_2) = \{< I, J >, < H, J >\}$ et que l'on souhaite retirer les noeuds et arêtes alors on a le graphe G_5 à droite dans la Figure 4.5.

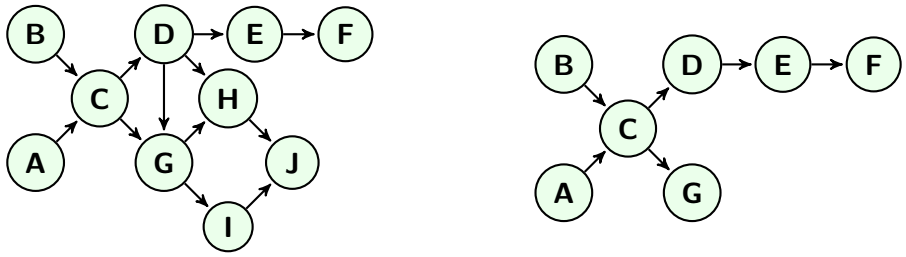


FIGURE 4.5 – Deux modèles de filière G_4 et G_5

4.4 Données synthétiques

Dans cette sous-section, nous exposons une expérimentation sur des données synthétiques générées à partir d'un jeu de simulation de réseau manufacturier. Dans un premier temps, nous expliquons le contexte et le principe général du jeu. Puis nous montrons le protocole expérimental qui nous a permis d'appliquer notre méthode pour extraire des anomalies que nous avions injectées dans le jeu de données en vue de les retrouver. Finalement, nous montrons les résultats quantitatifs de notre méthode et démontrons que son temps d'exécution est linéaire en fonction de plusieurs paramètres comme la taille de la base de données, le nombre d'attributs, ou encore la taille du modèle de filière.

4.4.1 La simulation d'une industrie manufacturière avec OpenTTD

Pour valider notre méthode quantitativement nous avons choisi d'utiliser un générateur de données synthétiques. Cependant, notre objectif est d'avoir des données synthétiques cohérentes et la génération de traces aléatoires n'est pas pertinente. Plusieurs structures de données doivent être construites : le modèle de filière, représentant le système manufacturier, un ensemble de traces unitaires servant à la génération de contextes.

Le jeu OpenTTD (Open Transport Tycoon Open Source) [8] est un jeu de gestion de fret et de construction d'un réseau de transport complet. L'intérêt principal est qu'il prend en compte l'intégralité d'un cycle de vie des ressources transportées (création, transport, vente, stockage) et que ces données peuvent être récupérées. Au cours du temps, le paquet de ressources peut être fragmenté pour que les éléments soient vendus à des endroits différents. La trace unitaire est donc constituée d'un ensemble de séquences symbolisant les séquences de positions pour chaque fragment de ressources au cours du temps. De plus, le jeu simule des pannes, des accidents ou des catastrophes, ce qui est réaliste par rapport à notre cas d'application réel. Nous avons développé un script au sein du jeu permettant de récupérer les évènements et de les stocker sous une forme de traces unitaires. La flexibilité du jeu nous a permis de construire un modèle de filière présenté dans la figure 4.6 pour construire des jonctions entre les différentes stations. Les ressources sont produites sur les deux stations en bas à gauche de la carte pour être acheminées le long du système manufacturier automatiquement dans des véhicules. Le processus de génération de données consiste alors à construire un réseau manufacturier dans le jeu et à laisser tourner le jeu qui va produire les traces correspondantes aux ressources. La capacité de production et le parcours dans le réseau manufacturier sont dépendants des contraintes internes du jeu, ainsi pour produire plus de traces unitaires il suffit de produire un réseau manufacturier plus grand (avec plus d'usines par exemple) et/ou de laisser tourner le jeu plus longtemps. Nous pouvons nous servir de scripts pour ajouter dans les données de traçabilité des simulations de détournements de produits et ainsi voir si notre méthode les découvre correctement. Lorsque le jeu est lancé, celui ci génère de très nombreux évènements pour chaque paquet de ressources. Nous procédons à un traitement des données qui permet de ne conserver que les évènements de passages d'une station à une autre. Dans nos expériences, nous avons ensuite dupliqué ce jeu de données un certain nombre de fois pour déterminer si notre méthode passe à l'échelle selon plusieurs paramètres (nombre d'attributs, taille du modèle de filière, nombre d'instances).

4.4.2 Protocole expérimental

Nous avons utilisé le modèle de filière décrit par la Figure 4.6. Celui ci est composé de 10 noeuds, et de 10 arêtes. Après avoir lancé le jeu pour produire des évènements, nous avons transformé ces données sous forme de traces unitaires, où une trace est égale à un paquet de ressources. Par exemple, un paquet peut être un ensemble de dix unités de bois produit dans une usine et qui est acheminé à divers endroits (des stations). A chaque station, le paquet peut être fragmenté en plusieurs ensembles qui suivront des chemins différents. Nous considérons une trace comme la totalité des déplacements effectués par un ensemble de ressources créées en même temps. Une séquence de sites ($trajectory(t)$), nécessaire pour calculer l'anormalité d'une trace, est donc un ensemble de séquences de sites décrivant les déplacements de chaque ressource d'un paquet. Ainsi, on calcule pour chaque séquence de sites de la trajectoire, le score d'anormalité comme suit :

8. http ://www.openttd.fr/

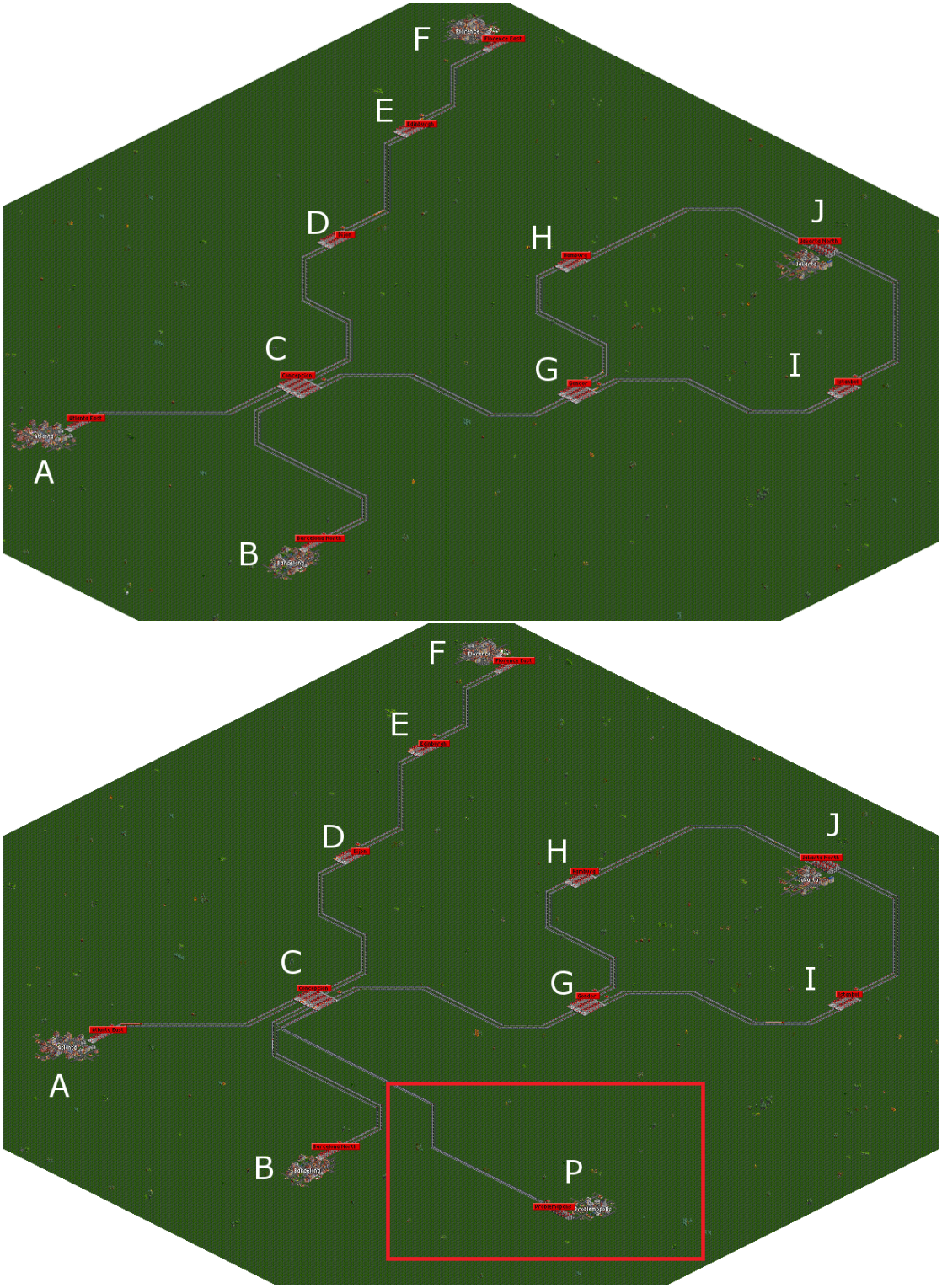


FIGURE 4.6 – Deux modèles de filière encodés dans le jeu OpenTTD, avec, en haut, le modèle de référence, et en bas le modèle pour le jeu de données 2

Definition 50 (Score d'anormalité pour un paquet)

$$\mu(t, M) = \frac{\sum_{i=1}^{i=n} \mu(trajectory_i(t), M)}{n}$$

soit $trajectory_i(t)$ la *ième* séquence de sites de la trace t parmi n trajectoires et M la matrice d'anormalité. Le score d'anormalité est alors la moyenne du score d'anormalité de chaque séquence de sites construits à partir de la trace.

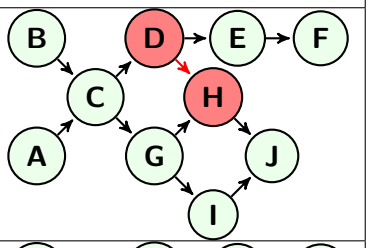
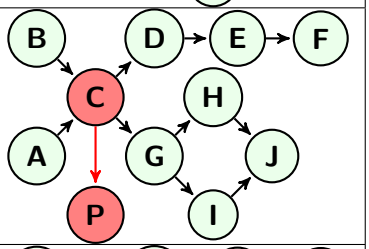
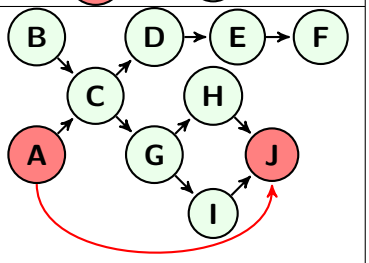
# du jeu	$\lvert\mathcal{D}^-\rvert$	$\lvert\mathcal{D}^+\rvert$	$\lvert\mathcal{I}\rvert$	Erreur insérée
1	22	823	319	
2	43	1449	462	
3	109	1826	836	

TABLE 4.1 – Caractéristiques des 3 jeux de données et la description des erreurs insérées

Pour nos expériences, nous travaillons sur trois jeux de données synthétiques où nous avons inséré une anomalie différente dans chacun. La table 4.1 synthétise l'ensemble des trois jeux de données utilisés. Les trois jeux possèdent comme descriptions le type de marchandises (passenger, mail), la liste des sites ainsi que les jours durant lesquels voyagent les produits. Le nombre de transactions dans chaque base, ainsi que le nombre de propriétés booléennes générées est indiqué dans la table 4.1. Chaque jeu de données a été généré suivant le protocole suivant : nous lançons la génération de données avec le modèle de référence initial pendant quelques secondes. Puis ensuite, nous continuons de générer des données en simulant un problème grâce à d'autres modèles de filière présentés dans la table 4.1 pendant un court instant. Le premier jeu de données a été généré grâce au modèle de filière présenté dans la dernière colonne de la table 4.1.

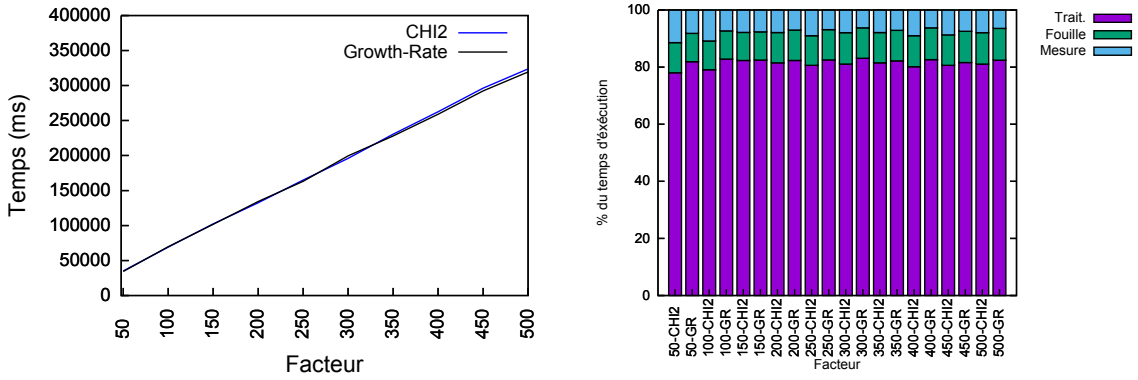
Nous avons délibérément introduit une transition anormale entre D et H qui est non présente dans le modèle de filière de référence. Pour cette expérience, nous connaissions l'anomalie insérée. Dans les deux autres expériences, nous avons demandé à quelqu'un d'insérer deux anomalies différentes sans en informer l'analyste, et nous avons cherché à les découvrir.

4.4.3 Résultats quantitatifs

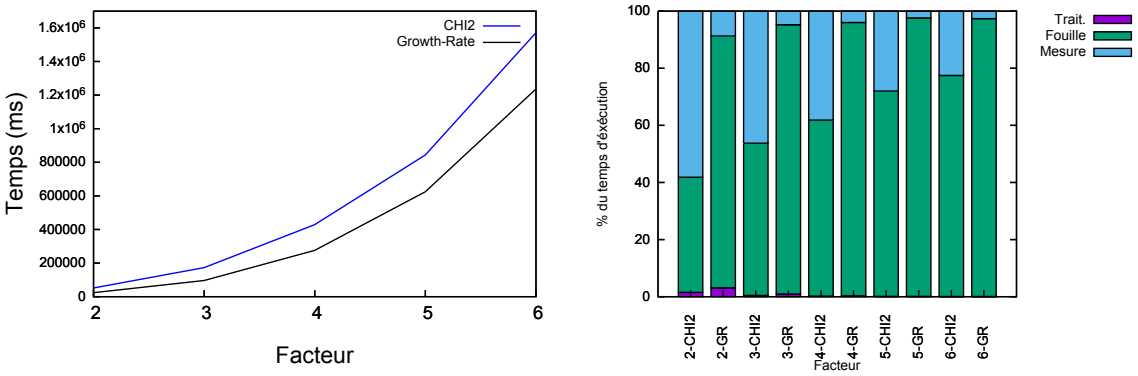
Nous cherchons dans un premier temps à déterminer l'évolution du temps d'exécution de notre méthode en fonction de plusieurs paramètres : le nombre d'instances de traces, la taille du modèle de filière en nombre de noeuds et d'arêtes, et le nombre de propriétés booléennes. Dans toutes les expérimentations, nous prenons les valeurs de paramètres suivantes $min\_sup = 0.001\%$ et $\theta = 0.04\%$. Dans cette première expérience, nous appliquons notre méthode, avec les mesures $\mathcal{X}^2$ et $Growth - Rate$, sur le premier jeu de données synthétique composé de 845 traces unitaires et 319 propriétés booléennes (sites, jours et type de marchandises). Notre méthode, pour analyser le passage à l'échelle, est de dupliquer ces données selon un facteur. Dans un premier temps, nous faisons varier la taille de la base en multipliant le nombre d'instances par un facteur croissant : x50 (42250 traces), x100 (84500 traces), ..., jusqu'à x500 (422500). Nous voyons sur le graphe de gauche sur la Figure 4.7 que les deux mesures ont des temps d'exécution linéaires similaires. Nous voyons dans le graphique de droite la proportion du temps d'exécution pour chaque étape de notre méthode (construction du modèle de filière et partitionnement des bases, fouille de motifs, puis calcul des mesures). On remarque que le temps de traitement occupe une grande partie du processus et que la proportion reste stable en fonction du nombre d'instances. En effet, quand celui-ci augmente alors chaque étape prend proportionnellement plus de temps. Si le temps de traitement est long, il n'est pas nécessaire en revanche de refaire cette opération à chaque fois. En effet, la séparation des traces se fait à partir des séquences de sites (les trajectoires), ainsi si l'on change les descriptions (c'est-à-dire que l'on reconstruit des contextes de fouille différents), il n'y a pas besoin de refaire cette étape de traitement. En revanche, si le modèle de filière est mis à jour, il est nécessaire de retraiter la base entière. Dans la deuxième expérimentation, nous avons appliqué notre méthode sur la même base de traces unitaires, mais en multipliant cette fois ci le nombre de propriétés booléennes de chaque description par 2, 3, 4, 5 puis 6. On observe, sur le graphique (b) que le temps d'exécution augmente de façon exponentiel. En effet, le nombre de motifs possibles à explorer devient vite très important. On remarque que le temps de calcul de la mesure est élevé au début, puis se réduit, par rapport au temps d'exécution de la fouille de motifs.

4.4.4 Résultats qualitatifs

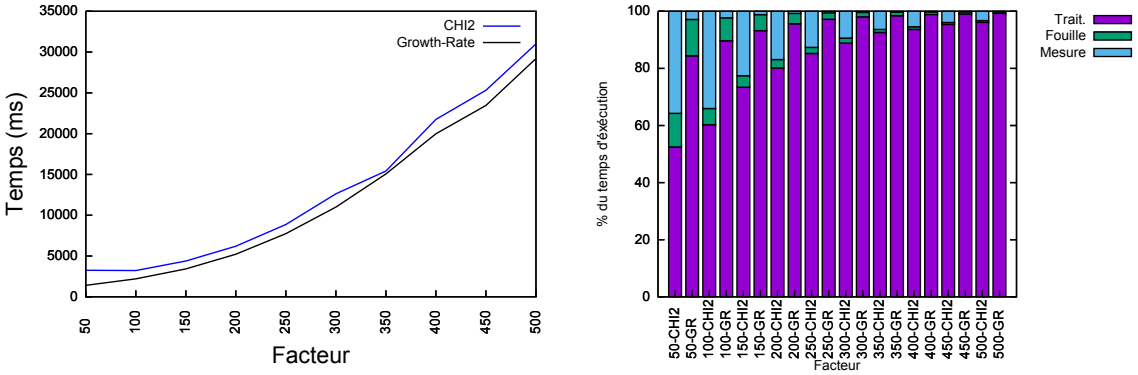
Dans nos expériences sur des données synthétiques, nous avons établi le réseau des comportements normaux comme décrit dans la première capture d'écran en haut de la Figure 4.6. Les graphes correspondants pour chaque anomalie sont montrés dans la Table 4.1. Les flèches noires montrent les transitions autorisées entre deux sites et les



(a) Mutiplication du nombre d'instances



(b) Mutiplication du nombre de propriétés



(c) Mutiplication de la taille du modèle de filière

FIGURE 4.7 – Evolution du temps d'exécution (en ms) en fonction d'un facteur de multiplication du nombre d'instances initiales (a) (x50, x100, ..., x500), du nombre de propriétés (b) et du nombre de noeuds et d'arêtes du modèle de filière (c).

flèches rouges montrent les transitions non autorisées, que nous avons inséré dans chacun exemple. Pour insérer des anomalies synthétiques dans nos jeux de données de manière réaliste, nous avons simulé deux anomalies : dans le premier jeu de données, nous avons fait comme si des marchandises était acheminée entre Hambourg (site H) et Dijon (site D) alors que cette transition n'est pas présente dans le réseau manufacturier décrivant les transitions autorisées. Cette anomalie simule un détournement de produits destinés à certains sites (par exemple certains pays uniquement), mais qui sont détournés pour une autre destination (un autre distributeur ou point de vente par exemple) qui n'est pas attendue par les fabricants des produits. Nous ne connaissons pas le pourcentage de traces anormales, puisque nous avons uniquement décrit, via le jeu, la liaison anormale, puis nous avons lancé la génération de données. Dans le deuxième jeu de données, nous avons crée un site non présent dans le modèle de filière puis nous avons simulé un détournement depuis le site C (Conception) vers ce site (Problemopolis). Pour évaluer notre méthode, nous ne pouvons tester si les produits jugés anormaux le sont effectivement, puis le jeu ne peut pas enregistrer une telle information. Il s'agit en revanche de voir si les motifs les plus anormaux retournés sont des descriptions cohérentes avec les anomalies insérées. Nous avons généré un contexte de fouille simple contenant le type de marchandises (des passagers ou du courrier), les jours durant lesquels voyagent les marchandises et l'ensemble des localisations.

#	Emergence	Support	Motif
1	-0.98	25	$\{mail, Dijon, Hamburg, 27/2/2033, 18/3/2033,$ $12/3/2033, 6/4/2033, 5/3/2033, 25/3/2033, 31/3/2033,$ $13/4/2033, 19/4/2033, 26/4/2033\}$
2	-0.98	26	$\{mail, 18/3/2033, 25/3/2033\}$
3	-0.98	26	$\{Hamburg, 6/4/2033, 25/3/2033\}$
4	-0.98	26	$\{mail, Dijon, 18/3/2033, 5/3/2033\}$
5	-0.98	26	$\{Hamburg, 13/4/2033, 19/4/2033, 26/4/203326\}$
6	-0.98	27	$\{mail, Hamburg, 13/4/2033, 26/4/2033\}$
7	-0.98	27	$\{mail, 6/4/2033\}$
8	-0.98	27	$\{5/3/2033, 25/3/2033\}$
9	-0.98	27	$\{mail, 25/3/2033\}$
10	-0.98	27	$\{mail, 13/4/2033, 19/4/2033\}$

TABLE 4.2 – Les dix motifs les plus discriminants pour le premier jeu de données

Notre premier jeu de données comporte 845 traces et 319 propriétés booléennes. Nous fixons le seuil de fréquence à 0.001% pour récupérer l'ensemble des motifs discriminants. Nous fixons le seuil θ en fonction de la base de données négatives que nous obtenons, pour pouvoir avoir un nombre conséquent d'exemples négatifs. Nous choisissons le seuil $\theta = 0.003\%$, ce qui permet d'avoir les tailles $|\mathcal{D}^+| = 823$ et $|\mathcal{D}^-| = 22$. Nous utilisons la mesure $\mathcal{X}^2$ pour extraire les motifs, et nous obtenons 4281 motifs caractéristiques d'anomalies. Nous présentons dans la Table 4.2 les 10 motifs ayant la valeur $\mathcal{X}^2$ normalisée la plus proche de -1. Dans le cas de notre jeu de données synthétiques, nous sommes dans un cas

idéal puisqu'il n'y a pas d'erreurs d'enregistrements, et qu'il n'y a qu'une anomalie, ce qui permet d'extraire des motifs caractéristiques d'anomalies avec une émergence presque égale à -1 (-0.98). Le motif 1 le plus anormal (et le moins fréquent) décrit bien notre anomalie : nous avons les deux sites (Dijon et Hambourg) puis d'autres informations : le type de marchandises (mail) et la liste des jours durant lesquels ces objets ont voyagé. On constate alors que l'anomalie insérée est bien décrite.

#	Emergence	Support	Motif
1	-1.0	43	$\{Concepcion, passenger, Problemopolis\}$
2	-0.65	522	$\{Concepcion, passenger\}$
3	-0.02	6	$\{Concepcion, passenger, Hamburg, Problemopolis\}$
4	-0.015	5	$\{Concepcion, passenger, Edinburgh, Problemopolis\}$
5	-0.015	5	$\{Concepcion, Dijon, passenger, Problemopolis\}$

TABLE 4.3 – Les cinq motifs les plus discriminants pour le deuxième jeu de données

Dans le deuxième jeu de données nous avons 1492 traces (avec $|D|^- = 43$ et $|D|^+ = 1449$) et 462 propriétés booléennes. Nous extrayons un total de 2842 motifs caractéristiques d'anomalies dont les 5 premiers motifs sont présentés dans la Table 4.3. Nous voyons que le premier motif décrit parfaitement toutes nos traces anormales : sa valeur d'émergence est de -1 et son support de 43 (égal à $|D|^-$). Nous voyons que l'anomalie concerne 43 passagers voyageant entre Conception et Problemopolis (notre site non présent dans le modèle de filière). Le deuxième motif possède une valeur d'émergence plus élevée (moins proche de -1) et a un support plus grand (522). Les trois autres motifs ont des valeurs d'émergence très proches de 0 et concernent également le site non présent dans le modèle de filière. Nous constatons une nouvelle fois que notre méthode capture bien les ensembles d'objets anormaux, et trouve leur description, tout en réduisant le nombre total de motifs caractéristiques d'anomalies. En effet, si on applique un seuil sur la valeur d'émergence, il est possible alors de ne conserver qu'une très petite partie des motifs les plus informatifs (par exemple avec un seuil comme -0.50, nous obtenons uniquement les deux premiers motifs au lieu de 2842).

Enfin, le troisième jeu de données contient un détournement entre A et J, qui est un chemin possible (en passant par A, C, G, I). La base comporte 1935 transactions et 109 sont classées comme négatives, et nous avons extrait 2930 motifs discriminants de la base négative. Comme nous le voyons dans la Table 4.4, le motif le plus anormal possède une valeur très proche de -1 (-0.97) et un support de 155. Sa description est $\{AtlantaEast, JakartaNorth\}$ ce qui correspond à l'anomalie qui avait été insérée par une tierce personne. Les motifs suivants donnent plus d'informations, notamment sur le type de ressources (ici des personnes ou du courrier).

4.5 Données réelles

Dans cette section, nous détaillons les expérimentations sur des données réelles du jeu vidéo Dota 2. Dans un premier temps, nous présentons le principe du jeu, puis nous

#	Emergence	Support	Motif
1	-0.97	155	$\{AtlantaEast, JakartaNorth\}$
2	-0.31	95	$\{passenger, AtlantaEast, JakartaNorth\}$
3	-0.27	192	$\{passenger, AtlantaEast\}$
4	-0.18	385	$\{passenger, Jakarta, North\}$
5	-0.16	60	$\{mail, AtlantaEast, JakartaNorth\}$
6	-0.13	149	$\{mail, AtlantaEast\}$
7	-0.11	230	$\{mail, JakartaNorth\}$
8	-0.03	27	$\{AtlantaEast, JakartaNorth, 4/11/2032, 11/11/2032\}$
9	-0.02	22	$\{4/11/2032, 11/11/2032\}$
10	-0.02	23	$\{AtlantaEast, 4/11/2032\}$

TABLE 4.4 – Les dix motifs les plus discriminants pour le troisième jeu de données

décrirons comment le graphe de référence a été conçu, avant de détailler les différents scénarios pour détecter les erreurs de stratégie.

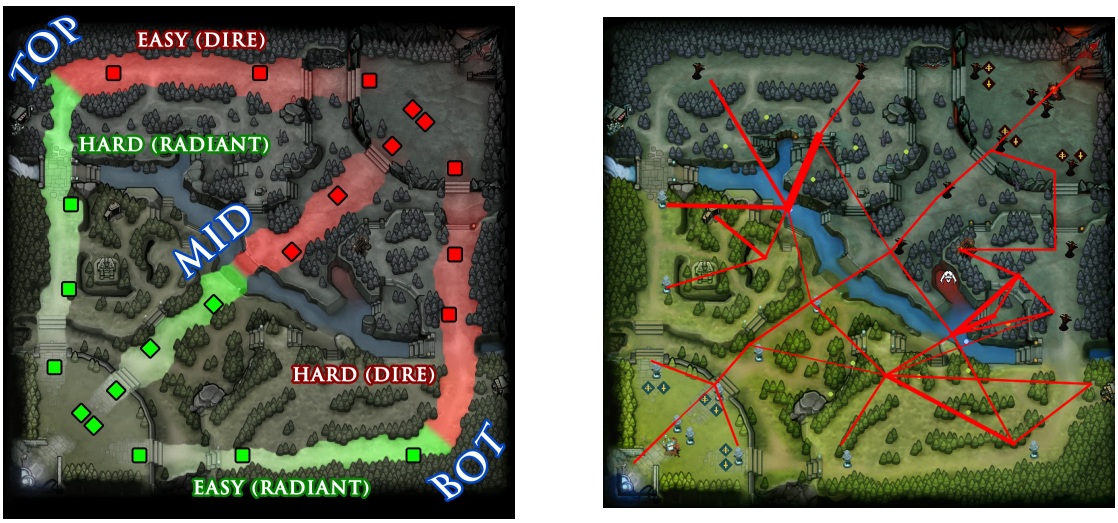


FIGURE 4.8 – Terrain de Dota2 à gauche et modèle $G_{darkseer}$ à droite

4.5.1 DOTA2 un jeu de stratégie en temps réel

Une partie est jouée sur un terrain de jeu où deux équipes de cinq joueurs s'affrontent en temps réel. Chaque équipe doit défendre son *château* et détruire celui de l'opposant pour gagner. Chaque joueur contrôle un héros qu'il peut déplacer sur le terrain (contrôle souris/clavier) et doit l'entrainer en collectant de l'argent, de nouveaux objets, des compétences et en se battant contre les forces ennemies. Dans la Figure 4.8, à droite, on peut voir les zones d'influence initiales des deux équipes. L'équipe rouge appelée *the dire* (resp. verte pour *the radiant*) défend son château situé au coin bas-gauche (resp. coin haut-

droit). Trois "chemins" principaux (*top*, *mid*, *bot*) séparent les équipes et contiennent des tours défensives. Chaque joueur a un rôle bien défini, qui dépend du héros qu'il a choisi (parmi 110 héros). Par exemple, un rôle consiste à défendre et étendre la zone d'influence sur un chemin spécifique ; un autre est de changer souvent de chemin pour rejoindre un allié et attaquer un ennemi par surprise. Sachant qu'une équipe ne peut voir que les zones qu'elle contrôle et donc estimer la position des ennemis, le fait de déclencher des attaques synchronisées est la clef du succès, tout en sachant garder son rôle initial sans quoi la progression du héros (amélioration de ses capacités) est beaucoup plus lente.

4.5.2 Construction d'un graphe de référence

De manière similaire à un jeu de rugby, le positionnement d'un héros sur le terrain est crucial pour tenir la ligne de défense. Rappelons qu'à chaque héros correspond un rôle : ce rôle n'est cependant pas indiqué aux nouveaux joueurs, et l'apprentissage du jeu est assez long. A chaque rôle correspondent aussi des zones optimales à contrôler pour avoir de meilleures chances de gagner. D'après les joueurs experts, chaque héros peut prétendre à au moins trois rôles parmi une quinzaine [9]. Nous modélisons donc le terrain de jeu par un graphe dont les sommets sont des points d'intérêt connus (château, tours, magasins, ... un sur-ensemble des points rouges et verts de la carte à gauche dans la Figure 4.8). Les arêtes correspondent aux transitions entre deux points d'intérêt, c'est-à-dire quand un héros est détecté à un point d'intérêt différent du point d'intérêt précédent dans la trace de comportement. Plus précisément, nous analysons toutes les traces, puis nous comptons chaque transition d'un point d'intérêt à un autre. Chaque arête est donc pondérée par le nombre de transitions entre chaque point d'intérêt observé dans toutes les traces de jeu. Finalement, nous calculons la moyenne totale de ces poids, puis nous retirons toutes les arêtes dont le poids est strictement inférieur à la moyenne des poids de toutes les arêtes.

On part du principe qu'ayant une base de traces de mouvements assez conséquente pour un héros/rôle particulier, le comportement fréquent devrait présenter la normalité, c'est-à-dire les zones qu'il doit contrôler. Les traces doivent être bien choisies de tel sorte qu'elles ne doivent pas porter uniquement sur des débutants mais sur un ensemble de joueurs dont on sait que le niveau de jeu est bon voir élevé (par exemple, l'ensemble des traces d'une compétition mondiale). Dans le cas contraire, le normalité construite via le modèle correspondra à des mouvements de joueurs débutants et seront non représentatifs de bonnes stratégies. En décrivant les traces par un ensemble de propriétés qui dénotent des éléments de stratégies, on suppose que les descriptions d'anomalies seront des stratégies qui sont inefficaces, générées par les joueurs en phase d'apprentissage. Un tel exemple de graphe de référence est donné par la carte à droite dans la Figure 4.8 pour le héros *Dark Seer* à partir de 500 traces de jeux issues du site `http://dotabank.com/`. Le motif décrira donc les erreurs stratégiques, erreurs d'autant plus fortes que la mesure d'anomalie est haute, et le support indiquera classiquement la fréquence d'apparition de cette anomalie.

9. `http://www.dota2.fr/apprendre/guides/guide-des-differents-roles`

4.5.3 Exprimer des scénarios pour détecter des erreurs de stratégie

Nous avons construit de multiples scénarios pour décrire des erreurs de stratégies validées par des experts du jeu avec qui nous étions en contact.

Anomalies expliquées par une erreur dans le choix des compétences. Nous détaillons le choix des propriétés pour créer le contexte de fouille à partir de l'ensemble des 500 traces de jeux. Durant la partie, chaque héros, en fonction d'actions précises, gagne de l'expérience et des niveaux. A chaque niveau, il choisit une compétence, parmi 3, sauf aux niveaux 6 et 11 où il peut en choisir une quatrième. Le choix d'un chemin dans cet arbre de compétences est stratégiquement très important et donne lieu à des guides (manière de choisir les compétences à un niveau précis pour maximiser ses possibilités de victoire). Nous codons alors une propriété booléenne de type '*La compétence n a été prise au niveau x*', qui indique qu'une compétence (1,2,3 ou 4) a été prise par le joueur à un niveau donné. De plus, nous introduisons une propriété qui indique s'il a acheté ou non un objet [10] très fréquemment pris par ce *héros*. Avec un seuil d'anomalie $\theta = 0.3$ (il faut que 30% des transitions du héros ne respectent pas le modèle pour que la trace soit considérée anormale) et un seuil de support minimum très bas $min\_sup = 1\% = 5$, 175849 motifs fréquents sont extraits, mais seuls 6 ont une mesure d'anomalie négative, c'est-à-dire des descriptions de traces qui s'écartent fortement du modèle. En partie listés dans la Table 4.5, ces motifs ont la même mesure d'anomalie -0.199 et un support très faible : il y a 5 instances dans l'image de chaque motif, nous faisons bien face à une anomalie. Selon notre expert, ces motifs montrent des erreurs de stratégies évidentes. Par exemple, choisir la compétence 4 au niveau 8 (propriété *comp_4_at_level_8*) est une erreur majeure, malgré tout faite dans 5.4% des 500 parties dont nous disposons [11]. On observe aussi qu'ils n'ont pas acheté l'objet "soul ring" ce qui est également rare. Nous avons retrouvé le profil des 5 joueurs impliqués dans le motif #1 au moment auquel les parties avaient été jouées : chacun n'avait réalisé qu'un nombre très faible de parties, ce sont bien des débutants.

Autres explications d'anomalies. En intégrant d'autres propriétés, établies après discussions avec notre expert du jeu, nous enrichissons le contexte de fouille et la manière d'expliquer une anomalie. Nous avons (i) un ensemble de propriétés qui indiquent les adversaires du héros dans la partie (chaque héros a des avantages/inconvénients face aux autres), (ii) le nombre de sbires (*creeps*) éliminés par le héros (les sbires apparaissent régulièrement sur le terrain, ne sont pas contrôlables, mais les éliminer rapporte de l'expérience au joueur), (iii) la réponse à *la compétence 4 est-elle prise au plus tôt ?* et enfin (iv) quand on observe que le nombre de passages aux bases est supérieur à un seuil anormalement haut (c'est-à-dire que le joueur est revenu souvent à la base pour se régénérer, la moyenne étant de 20 passages). Nous avons mis le seuil d'anomalie à 22% et le support minimum à 1% et nous obtenons 193026 motifs fréquents. Nous remarquons là encore qu'avec un seuil d'anomalie approprié le nombre de motifs émergents en sortie est très faible puisque nous obtenons 16 motifs émergents dont 9 sont présentés dans la table

10. Le *soul ring* `http://www.dotabuff.com/heroes/dark-seer/items`
11. http ://www.dotabuff.com/heroes/dark-seer/builds

4.6. On constate que parmi les héros adversaires du *Dark Seer* ceux-ci ont souvent gagné contre lui dans la plupart des matchs [12]. Le motif de la ligne 1 possède une propriété qui indique un très faible nombre de sbires tués (entre 20 et 30 alors que la moyenne est de 100 à 200). On observe alors plusieurs types d'anomalies : la première qui porte sur le nombre de *sbires* tué en co-occurrence avec la compétence 4 qui n'est pas prise au niveau 6. Les motifs de la ligne 4 et 5 concernent des joueurs étant souvent revenus à leur base et le motif 9 permet de voir que des joueurs n'ont pas réussi à pénétrer dans la base ennemie (*no_dire_fort*). Notre méthode permet de caractériser différents types d'anomalies à partir d'une même classification des traces.

#	Emergence	Support	Motif
1	-0.199	0.1	$\{dire, no\_soul\_ring\_item,$ $comp\_2\_at\_level\_1, comp\_2\_at\_level\_3,$ $comp\_4\_at\_level\_8, comp\_1\_at\_level\_10,$ $comp\_4\_at\_level\_11, comp\_3\_at\_level\_14\}$
2			$\{dire, no\_soul\_ring\_item,$ $comp\_2\_at\_level\_3, comp\_2\_at\_level\_7,$ $comp\_4\_at\_level\_8, comp\_1\_at\_level\_9,$ $comp\_1\_at\_level\_10, comp\_4\_at\_level\_11,$ $comp\_3\_at\_level\_14\}$
3			$\{dire, no\_soul\_ring\_item,$ $comp\_2\_at\_level\_1, comp\_3\_at\_level\_2$ $comp\_2\_at\_level\_3, comp\_2\_at\_level\_5,$ $comp\_1\_at\_level\_6, comp\_4\_at\_level\_8,$ $comp\_1\_at\_level\_9, comp\_4\_at\_level\_11,$ $comp\_3\_at\_level\_14\}$

TABLE 4.5 – 3 motifs, avec $\theta = 30\%$ et $min\_sup = 1\%$.

Sur le choix des seuils Nous utilisons deux paramètres : un support minimum *min_sup* ainsi qu'un seuil d'anomalie θ. Nous cherchons les motifs qui maximisent à la fois le support et le score d'anomalie, deux mesures antinomiques. Le choix du seuil d'anomalie θ impacte fortement le résultat. Il s'agit du pourcentage de transitions qui ne respectent pas le modèle : avec $\theta = 0$ la moindre transition non licite rend la trace anormale. La Figure 4.9 montre le pourcentage de traces assignées à la classe anormale en fonction de θ : augmenter θ c'est être plus permissif, il en résulte moins de motifs. En faisant varier θ, on peut trouver les descriptions d'anomalies les plus flagrantes. On voit que l'équilibre est atteint vers $\theta = 0.16$. Si on affiche un nuage de points des motifs avec leur valeur de support et leur mesure d'anomalie normalisée dans $[-1, +1]$, avec -1 quand le motif n'apparait que dans des instances de la classe négative, l'influence du seuil est encore plus évidente. Sur la figure du milieu, avec $\theta = 0.16$ on voit que la distribution est centrée autour de 0. En augmentant le seuil θ la distribution est majoritairement dans la classe positive et très peu de motifs sont caractéristiques d'anomalies.

12. Voir la page `www.dotabuff.com/heroes/dark-seer/matchups`

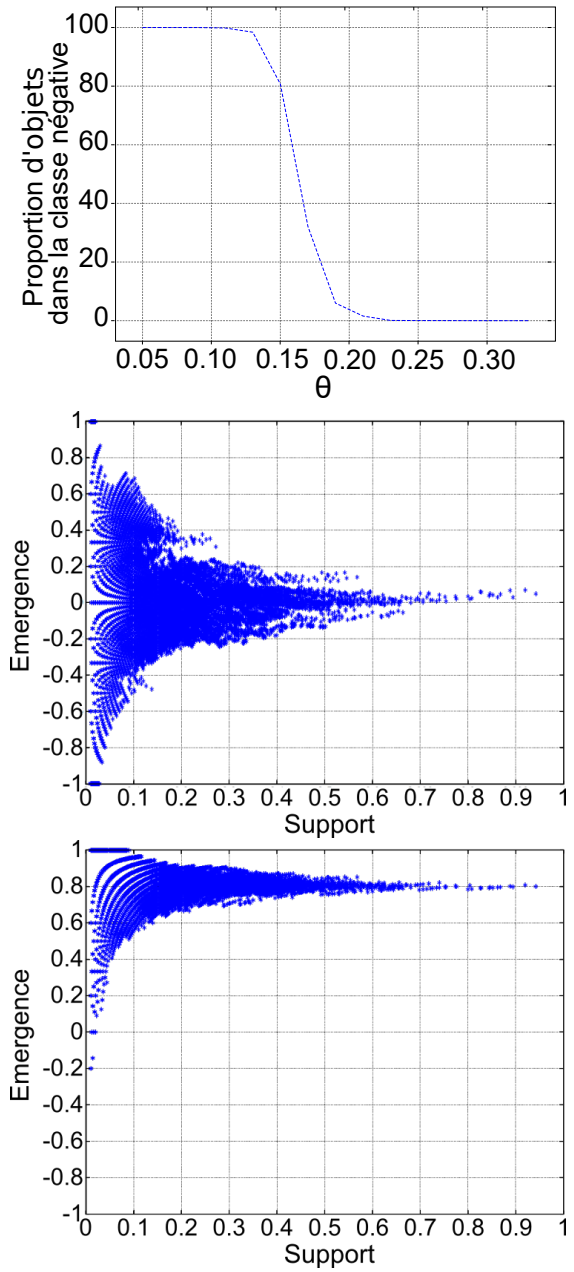


FIGURE 4.9 – Nombre de motifs émergents en fonction de θ en haut à gauche, et distribution des motifs selon leur support/score d'anomalie pour $\theta = 0.16$ (en haut à droite) et $\theta = 0.24$ (bas)

#	Emergence	Support	Motif
1	-0.11	0.018	$\{creeps\_killed\_between\_20\_and\_30, no\_comp\_4\_level\_6\}$
2	-0.14	0.014	$\{enemy\_lifestealer, enemy\_keeperofthelight, no\_comp\_4\_level\_6\}$
3	-0.16	0.024	$\{enemy\_riki, no\_comp\_4\_level\_6\}$
4	-0.19	0.01	$\{no\_comp\_4\_level\_6, > \_40\_dire\_fountain\}$
5			$\{no\_comp\_4\_level\_6, > \_40\_radiance\_fountain\}$
6			$\{enemy\_medusa, no\_comp\_4\_level\_6\}$
7			$\{enemy\_chen, enemy\_gyrocopter\}$
8			$\{enemy\_queenofpain, enemy\_gyrocopter\}$
9			$\{enemy\_bountyhunter, no\_comp\_4\_level\_6, no\_dire\_fort\}$
10			$\{enemy\_lifestealer, enemy\_keeperofthelight, no\_comp_4\_level\_6, no\_dire\_fort\}$
11	-0.25	0.016	$\{enemy\_furion, poi\_infrequent\_bot\_shop\}$
12	-0.33	0.012	$\{> \_40\_dire\_fountain\}$
13	-0.42	0.014	$\{enemy\_nyxassassin, poi\_infrequent\_bot\_shop\}$
14			$\{enemy\_rubick, no\_comp_4\_level\_11\}$
15	-0.60	0.01	$\{enemy\_nyxassassin, no\_comp_4\_level\_6, poi\_infrequent\_bot\_shop\}$
16	-0.66	0.012	$\{enemy\_queenofpain, no\_comp_4\_level\_11\}$

TABLE 4.6 – 16 motifs, avec $\theta = 22\%$ et $min\_sup = 1\%$.

4.6 Conclusion

La caractérisation de collections de traces unitaires anormales est importante dans le contexte manufacturier. En effet, elle permet de déduire des hypothèses sur les contextes dans lesquels surviennent des anomalies diverses : détournements, marchés parallèles, vols. Nous avons proposé dans ce chapitre, une méthode qui utilise une connaissance experte, sous la forme d'un graphe orienté appelé modèle de filière, pour déterminer le degré d'anormalité des traces unitaires. Intuitivement, nous considérons que plus le comportement, codé à partir des traces, d'un produit s'écarte de ce qui est décrit par les analystes (ou construit automatiquement pour produire un modèle de référence), plus la trace unitaire est anormale, puisqu'elle contrevient à de nombreuses règles en vigueur dans le système manufacturier (fixée par contrat par exemple). La caractérisation de ces ensembles de traces unitaires, par une classe positive (trace normale) ou négative (trace anormale), nous permet d'appliquer une méthode de fouille de motifs bien connue : l'extraction de motifs émergents [25]. Nous montrons dans nos résultats l'influence des différents seuils, de fréquence et d'anormalité, sur le nombre de résultats et sur les temps d'exécution. En effet, le seuil d'anormalité permet d'influer sur la taille de la base de traces négatives, puisque plus il est haut et plus le comportement d'une trace unitaire doit diverger du modèle pour être assignée à cette base, ce qui réduit sa taille et par extension le nombre de motifs. Cette méthode a été expérimentée sur des données synthétiques, mais réalistes, issue d'un jeu de simulation d'un réseau manufacturier, ainsi que sur des données réelles de traces comportementales issues d'un jeu vidéo.

Chapitre 5

Identification d'agents usurpant une identité

5.1 Introduction

Chaque site du système manufacturier est contrôlé par un agent logistique. Un agent peut contrôler un ou plusieurs sites tout au long de la chaîne de production. Entre chaque site, chaque produit est transporté par camion, bateau ou avion et il est possible alors que certains agents détournent des produits en vue de les copier et les revendre à un prix élevé en utilisant la notoriété du fabricant d'origine. Pour se prémunir de ces vols, exclure les agents des systèmes manufacturiers n'est pas suffisant car ils peuvent réapparaître sous d'autres identités et continuer à détourner des produits. Les agents, par exemple des vendeurs, génèrent des données comportementales, qui peuvent être partielles, lorsqu'ils transforment ou transportent des produits. Nous faisons l'hypothèse que quand un agent préalablement exclus réapparait sous une autre identité, alors ses données comportementales peuvent permettre de détecter les deux identités de l'agent. Dans ce chapitre, nous considérons que la trace comportementale d'un agent est un ensemble d'attributs, numériques par exemple, calculés à partir d'un ensemble d'actions effectuées par cet agent. Par exemple, le nombre de ventes effectuées, le nombre de transitions effectuées par les produits manipulés par le vendeur, le nombre de pays en moyenne que traversent ces produits, etc.

Dans ce chapitre, nous proposons une méthode permettant d'extraire les paires d'agents (qui sont en réalité les mêmes) grâce à un processus en deux temps : nous classifions chaque agent à partir de traces comportementales composées de descripteurs spécifiques et nous construisons un modèle de prédiction sur ces traces. On analyse la matrice de confusion, qui sert d'ordinaire à évaluer le modèle, grâce à une analyse de concepts formels pour détecter si deux agents partagent une erreur de classification localisée donc un comportement semblable susceptible de montrer que ces deux agents sont un seul et même individu.

Notre but est alors de construire des collections de traces comportementales, composées de descripteurs définis selon le cas d'application, dérivées de ses traces unitaires,

pour chaque agent. Dans ce chapitre, nous entendons par *traces*, un vecteur numérique construit à partir des traces unitaires en vue d'une utilisation dans le domaine de l'apprentissage automatique.

Soit un ensemble de traces labellisées par le nom des agents que l'on veut étudier, on peut construire un modèle de prédiction qui se chargera de déterminer les comportements caractéristiques de chaque agent et ainsi pouvoir l'identifier à partir de ses traces comportementales. On constate alors que nous avons la notion d'agent, qui désigne un ensemble d'individus ou un individu réel, et la notion d'avatar qui va indiquer l'identité affichée au sein même du système. Ainsi, un agent peut avoir un et un seul avatar mais celui ci peut également avoir plusieurs avatars s'il souhaite cacher sa véritable identité.

Le champ du jeu vidéo a donné lieu à des publications proposant de reconnaître les identités de joueurs à partir de leurs comportements. Yan et al. [92] ont construit un modèle de prédiction chargé de reconnaître les joueurs de Starcraft 2 à partir de leurs traces comportementales encodées avec des descripteurs indiquant les raccourcis claviers utilisés et le nombre de fois qu'ils sont utilisés. Ils ont démontré que la classification de traces de jeu est très efficace sur les données de début de partie (300 premières secondes). Cependant, leur jeu de données ne possède pas de joueurs avec des identités multiples sachant qu'ils les retirent au préalable (un agent possède donc un unique avatar). En effet, ils ont démontré que l'efficacité de classification des avatars chutait fortement lors de la présence de ces doublons, puisque les modèles ne sont pas capables de différencier les différents avatars d'un unique agent.

Dans le cas d'application de Yan et al. [92] comme dans les systèmes manufacturiers, il n'est pas possible de savoir quels sont les avatars contrôlés par un agent. Nous avons uniquement accès aux traces comportementales produites par un avatar.

Notre modèle peut être séparé en deux sous-tâches : nous construisons un modèle de prédiction à partir des traces comportementales des avatars, puis nous générons la matrice de confusion, et à partir de celle-ci nous appliquons un procédé de fouille de concepts formels pour découvrir les avatars qui serait susceptible d'être le même agent.

Notre hypothèse est que si un agent possède deux avatars, alors le classifieur 'hésitera' entre ces deux avatars, autrement dit il assignera de manière uniforme, selon le classifieur, les traces de chaque avatar à l'un et à l'autre. Nous détectons ces motifs particuliers issus de la matrice de confusion grâce à l'analyse de concepts formels et plus particulièrement avec les structures de patrons, qui permettent de prendre en considération des données numériques.

Dans un premier temps, nous détaillons les définitions du champ de l'analyse de concepts formels puis nous exposerons notre méthode de reconnaissance des avatars dupliqués point par point. Enfin, nous présenterons une évaluation de la méthode avec des données de jeu réelles en utilisant différents classifieurs, avant de conclure.

5.2 L'analyse de concepts formels

L'analyse de concepts formels permet de classifier des ensembles d'objets qui partagent un maximum d'attributs binaires sous la forme de concepts formels (Ganter et

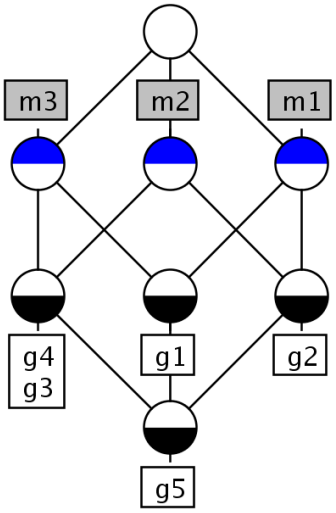


FIGURE 5.1 – Treillis de concepts formels issu de la table 5.1

	m_1	m_2	m_3
g_1	×		×
g_2	×	×	
g_3		×	×
g_4		×	×
g_5	×	×	×

TABLE 5.1 – Exemple d'une table binaire avec 5 objets et 3 attributs.

Wille [30]). Ces concepts sont ordonnés pour former un treillis, qui sert de support à la découverte de connaissances. Quand les objets sont décrits de manière complexe (nombres, intervalles, graphes, séquences,...), ces descriptions sont transformées en ensembles d'attributs binaires. Dans un premier temps, nous montrons l'approche classique de l'analyse de concepts formels, puis nous introduisons les structures de patrons qui permettent de prendre en charge directement des structures complexes telles que des vecteurs d'intervalles.

Soit G un ensemble d'objets, M un ensemble d'attributs et I une relation binaire définie sur le produit cartésien $G \times M$. Le triplet (G, M, I) est appelé un contexte formel. Pour un objet $g \in G$ et un attribut $m \in M, (g, m) \in I$ signifie que l'objet g possède l'attribut m. Par exemple, dans la Figure 5.1, l'objet g_1 possède les attributs m_1 et m_3. Dans notre cas d'application, chaque objet g est un agent qui représente l'identité d'un joueur (son avatar). Un attribut m représente une propriété booléenne qui peut indiquer si le joueur, tout au long de ses parties, va vérifier certaines contraintes en jouant son avatar (par exemple que son taux de victoire est dans un certain intervalle, que l'or gagné est en dessous d'un certain seuil, qu'il a pris certains héros ou pas, etc.)

Nous pouvons définir deux opérateurs de dérivations $(.)'$:

Definition 51 (Opérateurs de dérivations) *Étant donné $A \subseteq G$ et $B \subseteq M$, on définit :*

$$A' = \{m \in M \mid \forall g \in A \subseteq G : (g, m) \in I\}$$
$$B' = \{g \in G \mid \forall m \in B \subseteq M : (g, m) \in I\}$$

Ces opérateurs définissent une connexion de Galois entre l'ensemble des parties de G, noté 2^G et l'ensemble des parties de M noté 2^M. Le premier opérateur permet d'associer à un ensemble d'objets A l'ensemble des attributs que les objets de A possèdent en commun. Inversement, le second opérateur associe un ensemble d'attributs B à l'ensemble des objets qui possèdent au minimum tous les objets de B. Un concept formel est alors défini ainsi :

Definition 52 (Concept Formel) *Une paire (A, B) est un concept formel si :*

$$A' = B, B' = A$$

Par exemple, à un ensemble $A = \{g_3, g_4, g_5\}$ correspond l'ensemble $A' = \{m_2, m_3\}$ et à l'ensemble $B = \{m_1, m_2\}$ correspond l'ensemble $B' = \{g_2, g_5\}$. On appelle A l'extension du concept et B son intension. Un concept formel correspond, dans notre cas, à un ensemble maximal de joueurs partageant un nombre maximum de propriétés. L'ensemble de tous les concepts formels du contexte (G, M, I) est ordonné par la relation d'ordre suivante :

Definition 53 (Ordre partiel de l'ensemble des concepts formels) *Soit (A_1, B_1) et (A_2, B_2) deux concepts formels :*

$$(A_1, B_1) \leq (A_2, B_2) \iff A_1 \subseteq A_2 (B_2 \subseteq B_1)$$

On dit que (A_1, B_1) est un sous-concept de (A_2, B_2). L'ensemble de ces concepts ordonnés forme un treillis complet appelé treillis de concept du contexte (G, M, I). La figure 5.1 montre le treillis de concepts tiré de la table 5.1. Cet affichage ne montre chaque objet et attribut qu'une fois. Chaque nœud est un concept, son extension (resp. son intension) est composé de l'ensemble des objets (resp. attributs) de son nœud ainsi que de ceux des sous-concepts (resp. super-concepts). Par exemple, le nœud avec l'extension $\{g_4, g_3\}$ possède aussi l'objet de son sous-concept g_5 et les attributs m_3 et m_2. On constate ainsi qu'un concept formel correspond à un itemset fréquent et clos décrit précédemment.

Cette méthode prend en compte uniquement les attributs binaires, pour utiliser directement d'autres structures de données dans la construction des concepts. Il est donc nécessaire de généraliser les treillis de concept en prenant en compte d'autres structures plus complexes, à l'aide d'une structure de patrons ou encore en effectuant des transformations binaires (que nous exposerons dans le chapitre suivant).

Les structures de patrons La connexion de Galois vue précédemment met en relation des éléments du treillis des objets avec celui des attributs et inversement. Ces deux ensembles les objets $(2^G, \subseteq)$ et les attributs $(2^M, \subseteq)$, sont partiellement ordonnés, ce qui fait que pour utiliser des descriptions d'objets avec des attributs non binaires (numériques, intervalles, séquences, etc.), nous devons définir un ordre partiel pour ces descriptions en vue d'extraire des concepts. C'est l'idée de Gunter et Kuznetsov [31] formalisant des objets G possédant des descriptions dans D appelées *patrons*. Ces patrons sont ordonnés au sein d'un infinimum demi-treillis $(D, \sqcap)$ où $\sqcap$ est un opérateur de similarité. De plus, la fonction $\delta : G \to D$ permet d'associer à tout objet de G sa description $\delta(g)$ de G. Une *structure de patrons* est un triplet $(G, (D, \sqcap), \delta)$. Les éléments de D sont ordonnées selon un opérateur de subsomption $\sqsubseteq$:

Definition 54 (Ordre partiel de l'ensemble des patrons) *Soit c, d, les éléments de D sont ordonnés par la relation de subsomption $\sqsubset$ suivante :*

$$c \sqsubseteq d \iff c \sqcap d = c$$

Si on prend l'ensemble des attributs $M = \{m_1, m_2, m_3\}$ on a $\{m_1, m_2\} \subseteq \{m_1, m_2, m_3\}$ $\iff \{m_1, m_2\} \cap \{m_1, m_2, m_3\} = \{m_1, m_2\}$, l'intersection ensembliste possède les propriétés de l'infinium $\sqcap$. Un patron est alors un élément d'un infinimum demi-treillis. De la même façon que précédemment, il faut définir une connexion de Galois qui permet d'établir une relation entre les éléments du treillis des objets $(2^G, \subseteq)$ et les éléments du treillis des patrons $(D, \sqcap)$, définie de la manière suivante :

Definition 55 (Connexion de Galois entre les éléments du treillis d'objets)

$$A^{\square} = \bigsqcap_{g \in A} \delta(g)$$
$$d^{\square} = \{g \in G | d \sqsubseteq \delta(g)\}$$

avec g un objet de $A \subseteq G$, $\delta(g)$ une description de g et $d \in (D, \sqcap)$. Ainsi, le premier opérateur de la connexion de Galois permet de retourner une description d'un ensemble d'objets représentant la similarité de leurs descriptions, grâce à l'opérateur de similarité $\sqcap$. De manière duale, le second opérateur permet d'accéder au plus grand ensemble des objets dont la similarité est représentée par cette fonction. De la même façon que précédemment, les concepts de la structure de patrons sont des couples (A, d) avec $A \subseteq G$ et $d \in (D, \sqcap)$ avec $A^{\square} = d$ et $A = d^{\square}$ et sont ordonnés par $(A_1, d_1) \leq (A_2, d_2) \iff A_1 \subseteq A_2 (d_2 \subseteq d_1)$. Cet ensemble de concepts forment une structure de patrons.

Cette définition générique permet de travailler avec des descriptions variées, par exemple les intervalles. On peut définir alors un opérateur de similarité $\sqcap$ donnant l'infinimum d'un ensemble d'intervalles (présentée dans Kaytoue et al.[50]) :

Definition 56 (Opérateur de similarité pour des intervalles) *Soit deux intervalles* $[a_1, b_1]$ *et* $[a_2, b_2]$*, avec* $a_1, b_1, a_2, b_2 \in \mathbb{R}$ *:*

$$[a_1, b_1] \sqcap [a_2, b_2] = [min(a_1, a_2), max(b_1, b_2)]$$

Ainsi l'infinimum de plusieurs intervalles est le plus petit intervalle qui les contient tous. Il est possible alors d'ordonner les intervalles selon un ordre partiel comme présenté précédemment :

Definition 57 (Ordre partiel de l'ensemble des intervalles) *Soit* c, d*, les éléments de* D *sont ordonnés par la relation de subsomption* $\sqsubseteq$ *suivante :*

$$\begin{aligned} & [a_1, b_1] \sqsubseteq [a_2, b_2] \\ \iff & [a_1, b_1] \sqcap [a_2, b_2] = [a_1, b_1] \\ \iff & [min(a_1, a_2), max(b_1, b_2)] = [a_1, b_1] \\ \iff & a_1 \leq a_2 \text{ et } b_1 \geq b_2 \end{aligned}$$

Ce qui signifie que les plus petits intervalles subsument les plus grands qui les contiennent. La figure 5.2 montre un demi-treillis où les patrons sont des intervalles. Par exemple, on voit que $[4,5] = [4,4] \sqcap [5,5]$ et $[4,5] \sqsubseteq [5,5]$ avec $[4,5] \sqcap [5,5] = [4,5]$. Les structures de patrons ont été utilisées pour traiter des vecteurs d'intervalles [51], des séquences [13], des arbres [57] ou des dépendances fonctionnelles [11].

Nous utilisons cette méthode dans notre méthode pour la détection d'agents qui ont usurpé une identité. Cette méthode interviendra pour extraire des concepts formels dont la description est un vecteur numérique issue de la matrice de confusion. Celle-ci est construite lors d'une première étape de classification des agents à partir de leurs traces.

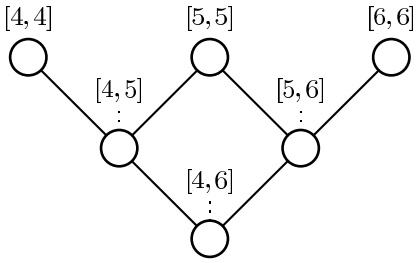


FIGURE 5.2 – Demi-treillis d'intervalles

5.3 Description de la méthode de reconnaissance des avatars dupliqués

Dans Cavadenti et al. [15], nous avons démontré que des descripteurs numériques spécifiques encodés à partir des traces unitaires et utilisés pour construire un modèle de prédiction ont donné de très bons résultats pour la détection d'avatars. Cependant, lorsque deux avatars sont générés par le même agent, l'efficacité de la classification chute fortement. Nous proposons une méthode pour détecter ces doublons en utilisant le résultat de la classification (matrice de confusion).

Soit A un ensemble d'avatars et T l'ensemble des traces comportementales telles que pour un avatar $a \in A$, l'ensemble $T_a \subseteq T$ est l'ensemble de toutes les traces générées par a. Un classifieur ρ permet, à partir d'une base de traces comportementales T, de construire un modèle prédictif cherchant à reconnaître un avatar a à partir d'une trace t. Dans le cas où le classifieur ρ reconnaît bien l'avatar a d'une trace, alors on a $\rho(t) = a$. Notre méthode consiste à utiliser la matrice de confusion générée par un classifieur ρ qui indique, pour chaque avatar, le nombre de traces qui ont été assignées à chaque avatar. Si, parmi les traces d'un avatar a_1, 10 ont été classifiées comme l'avatar a_1 et 10 comme l'avatar a_2, avec 0 traces classifiées comme d'autres avatars, alors le classifieur a réparti les traces entre ces deux avatars, ce qui permet de faire l'hypothèse que ces avatars sont en réalité contrôlés par le même joueur. Dans un premier temps, nous procédons à la classification de l'ensemble des traces en utilisant les avatars comme cibles. Notre but est d'obtenir des groupes d'avatars comme $\{a_1, a_2\}$ qui vont indiquer que ces deux avatars sont susceptibles d'être en réalité le même agent.

5.3.1 Classification des traces

Chaque agent, via son avatar, produit des traces comportementales issues de traces unitaires. Dans un premier temps, une étape d'étude des descripteurs pertinents, avec des méthodes de sélections automatiques de descripteurs, pour l'identification des avatars est nécessaire. Il s'agit de définir quels sont les attributs et les aspects des données présents dans les traces unitaires qui sont pertinents pour construire un modèle prédictif efficace. Un classifieur est une fonction $\rho : T \to A$ qui assigne un avatar $\rho(t) \in A$ à une trace donnée $t \in T$. Soit $n = |A|$ le nombre d'avatars dans A, pour n'importe quel classifieur ρ

appris à partir d'un ensemble de traces T, on peut dériver une matrice de confusion

$$C^{\rho}_{n\times n} = (c_{i,j})$$

où

$$c_{i,j} = |\{t \in T_{a_i} \ s.t. \ \rho(t) = a_j\}|$$

Chaque ligne et colonne de C^{ρ} correspond à un avatar, avec la valeur c_{ij} qui est le nombre de traces d'un avatar a_i qui sont classifiées par ρ comme un avatar a_j. La matrice de confusion normalisée est donnée par

$$\tilde{C}^{\rho} = [c_{i,j}/|T_{a_i}|]$$

où $\tilde{C}^{\rho}_{i,i} = 1$ pour chaque $i \in [1, |A|]$ qui signifie que toutes les traces de l'avatar a_i sont correctement classifiées par ρ. La table 5.2 donne un exemple d'une matrice de confusion normalisée de cinq avatars. Nous avons généralisé le classifieur à une fonction ρ et ainsi nous le traitons comme une boîte noire avec une entrée, qui est l'ensemble de traces T, et en sortie, pour chaque trace, la prédiction pour chaque avatar de A, sous la forme d'une matrice de confusion normalisée.

5.3.2 Regrouper les avatars

Une fois que l'on est en présence d'une matrice de confusion normalisée, il s'agit de définir les critères qui vont nous permettre de regrouper les avatars contrôlés par le même agent, que l'on nomment *doublons*. Notre intuition est que le classifieur va difficilement différencier les doublons, par conséquent, les valeurs de la matrice de confusion doivent être élevées et concentrées entre ces deux avatars uniquement. Dans la Table 5.2 nous voyons que le classifieur répartit les traces entre les avatars a_1 et a_2. Pour a_1, 60% des traces de a_1 sont bien reconnues, mais 40% sont reconnues comme étant celles de a_2. Inversement, 40% des traces de a_2 sont classifiées comme appartenant à a_1, 55% à a_2 et 5% à a_3. L'ensemble $\{a_1, a_2\}$ est alors un motif intéressant et susceptible de décrire une usurpation d'identités. De la même manière, $\{a_4, a_5\}$ peuvent être un autre agent contrôlant a_4 et a_5, tandis que a_3 est un singleton avec une valeur élevée (80%) lors de sa prédiction. Notre but étant de trouver les paires où le classifieur hésite entre les deux avatars de la paire, et uniquement entre ces deux avatars. Ainsi, un regroupement raisonnable des avatars est donné par le partitionnement suivant : $\{\{a_1, a_2\}, \{a_3\}, \{a_4, a_5\}\}$.

Plus formellement, soit une matrice de confusion normalisée $\tilde{C}^{\rho}$, nous voulons trouver les paires d'avatars $a_i, a_j \in U$ telles que $\tilde{C}^{\rho}_{ij} \simeq \tilde{C}^{\rho}_{ji} \simeq \tilde{C}^{\rho}_{ii} \simeq \tilde{C}^{\rho}_{jj}$ et $\tilde{C}^{\rho}_{ij} + \tilde{C}^{\rho}_{ji} + \tilde{C}^{\rho}_{ii} + \tilde{C}^{\rho}_{jj} \simeq 2$. Ces conditions surviennent quand, si a_i, a_j correspondent au même agent, les traces dans T_{a_i} ont la même probabilité d'être classifiée en a_i ou a_j (de même pour T_{a_j}). En outre, pour une trace d'un avatar a_i, il est nécessaire que la probabilité d'une classification soit répartie entre a_i et a_j seulement, ce qui signifie que $\tilde{C}^{\rho}_{ij} + \tilde{C}^{\rho}_{ii} \simeq 1$ (de même pour a_j). En utilisant cette hypothèse, nous proposons par la suite d'extraire les motifs de la matrice de confusion et les post-traiter pour obtenir des groupes de paires d'avatars candidats. La première étape est effectuée grâce à l'analyse de concepts formels [30, 31].

	a_1	a_2	a_3	a_4	a_5
a_1	0.6	0.4	0	0	0
a_2	0.4	0.55	0.05	0	0
a_3	0	0	0.8	0.15	0.05
a_4	0	0.05	0	0.7	0.25
a_5	0	0	0	0.5	0.5

TABLE 5.2 – Une matrice de confusion pour 5 avatars

La description de chaque objet A, ou avatar, est un vecteur numérique L^A où $L = [0,1]$ est un ensemble flou des degrés d'appartenance, c'est-à-dire, pour un avatar a_i, le pourcentage de ses traces qui ont été classées comme *appartenant* à a_i. Par exemple : pour a_1 on a l'ensemble $\{0.6, 0.4, 0, 0, 0\}$ ce qui signifie que lors de la classification, 60% des traces de l'avatar a_1 ont été classifiées comme les traces de a_1, 40% des traces comme l'avatar a_2 et 0% pour les trois autres avatars. Il est clair que le classifieur a alors "hésité" entre l'avatar a_1 et a_2. Pour assigner à chaque objet une description, on définit une fonction $\delta : A \rightarrow L^A$ qui assigne des valeurs d'appartenance pour un avatar a_i dans l'ensemble flou L^A basé sur la matrice de confusion normalisée formalisée précédemment. Cela revient à assigner à a_i la ligne correspondante dans $\tilde{C}^\rho$ que l'on note $\tilde{C}^\rho_i$. Nous modélisons alors $\tilde{C}^\rho$ comme une structure de patrons $(A, (L^A, \sqcap), \delta)$ où A sont les objets (avatars), $(L^A, \sqcap)$ les patrons (sous forme de vecteurs numériques) ordonnés sous la forme d'un infinimum demi-treillis avec un opérateur de similarité $\sqcap$ et la fonction δ qui assigne à chaque avatar une description. L'opérateur $\sqcap$ est un opérateur de similarité dans un demi-treillis (idempotent, commutatif et associatif), que nous définissons de la manière suivante :

Definition 58 (Opérateur de similarité pour une matrice de confusion normalisée) *Soit $a_i, a_j \in A$:*

$$\delta(a_i) \sqcap \delta(a_j) = \langle min(\tilde{C}^\rho_{ik}, \tilde{C}^\rho_{jk}) \rangle, k \in [1, |A|]$$
$$\delta(a_i) \sqsubseteq \delta(a_j) \iff \delta(a_i) \sqcap \delta(a_j) = \delta(a_i)$$

L'opérateur $\sqcap$ correspond à l'intersection d'ensembles flous, et le treillis $(L^A, \sqsubseteq)$ est un ordre partiel sur les éléments de L^A qui peuvent être représentés sous la forme d'un treillis de concepts que l'on peut construire comme dans l'ACF classique. La structure de patrons $(A, (L^A, \sqcap), \delta)$ est munie de deux opérateurs de dérivation, pour former une connexion de Galois [31] comme avec des attributs binaires. Formellement, nous avons :

Definition 59 (Connexion de Galois pour un sous-ensemble d'avatars) *Soit un ensemble d'avatars $\mathtt{A} \subseteq A$ et un ensemble flou $\mathtt{d} \in L^A$*

$$\mathtt{A}^\square = \prod_{a \in \mathtt{A}} \delta(a) \qquad \mathtt{d}^\square = \{a \in A \mid \mathtt{d} \sqsubseteq \delta(a)\}$$

Une paire $(\mathtt{A},\mathtt{d})$ est un concept si et seulement si $\mathtt{A}^{\square} = \mathtt{d}$ et $\mathtt{d}^{\square} = \mathtt{A}$. Les concepts sont ordonnés suivant l'inclusion de leur extension telle que pour $(\mathtt{A}_1,\mathtt{d}_1)$ and $(\mathtt{A}_2,\mathtt{d}_2)$ nous avons :

Definition 60 (Ordre partiel de l'ensemble des concepts)

$$(\mathtt{A}_1,\mathtt{d}_1) \leq (\mathtt{A}_2,\mathtt{d}_2) \iff \mathtt{A}_1 \subseteq \mathtt{A}_2 \, (ou \ \mathtt{d}_1 \sqsupseteq \mathtt{d}_2)$$

Intuitivement, un concept $(\mathtt{A},\mathtt{d})$ contient un ensemble flou $\mathtt{d}$ qui peut être représenté comme un *vecteur* $\mathtt{d} = \langle \mathtt{d}^j \rangle$ avec une taille $|A|$ où chaque valeur $\mathtt{d}^j$ est le minimum pour chaque ligne i de la colonne j pour la matrice $\tilde{C}^{\rho}$ avec $a_i \in \mathtt{A}$.

Exemple 11 *Soit la Table 5.2 qui présente la matrice de confusion de cinq avatars obtenue avec un classifieur ρ. Nous calculons l'intersection floue entre deux avatars a_1 et a_2 de la manière suivante :*

$$\begin{aligned}
\delta(a_1) &= \{a_1^{0.6}, a_2^{0.4}, a_3^0, a_4^0, a_5^0\} \\
\delta(a_2) &= \{a_1^{0.4}, a_2^{0.55}, a_3^{0.05}, a_4^0, a_5^0\} \\
\delta(a_1) \sqcap \delta(a_2) &= \{a_1^{0.4}, a_2^{0.4}, a_3^0, a_4^0, a_5^0\} \\
\{a_1, a_2\}^{\square} &= \{a_1^{0.4}, a_2^{0.4}, a_3^0, a_4^0, a_5^0\} \\
\{a_1, a_2\}^{\square\square} &= \{a_1, a_2\}
\end{aligned}$$

Alors $(\{a_1, a_2\}, \{a_1^{0.4}, a_2^{0.4}, a_3^0, a_4^0, a_5^0\})$ est un concept. Nous nous sommes basé sur l'algorithme Close-By-One [54] initialement prévu pour les concepts formels. Le principe de l'algorithme est d'effectuer une recherche en profondeur du treillis pour calculer les concepts. A chaque étape, l'algorithme étend le concept courant en ajoutant un objet dans l'extension, et en procédant à sa clôture. Un test de canonicité est effectué pour empêcher la redondance, c'est-à-dire en s'assurant que l'extension ne possède aucun objet qui précède l'objet courant. Nous montrons dans la Figure 5.3 le treillis des ensembles d'agents de chaque concept, ainsi que le score, qui permet de déterminer sa pertinence.

Pondérer les concepts et extraire les doublons Pour réduire le nombre de résultats, et ne garder que les concepts qui représentent des groupes d'avatars intéressants, nous avons développé la fonction de score suivante $s : L^A \to [0,1]$ pour un concept :

Definition 61 (Score pour un concept) *Pour un concept $\mathtt{d}$:*

$$s(\mathtt{d}) = \sum_{j=1}^{|A|} \mathtt{d}^j$$

Cette fonction s est décroissante suivant l'ordre des concepts $(\mathtt{A}_1,\mathtt{d}_1) \leq (\mathtt{A}_2,\mathtt{d}_2) \implies s(\mathtt{d}_1) \leq s(\mathtt{d}_2)$. Ainsi, les concepts peuvent être extraits suivant un seuil pour le score de la même manière qu'avec le seuil de support minimum dans la fouille de motifs [8]. En effet, si un motif (comme un ensemble fréquent) apparaît fréquemment dans les

données, on sait que la fréquence de ses super-motifs sera égale ou supérieure à celle du motif. De la même façon, le score ne peut que décroître puisque nous prenons à chaque fois les valeurs minimales lorsque nous calculons l'intersection floue de deux vecteurs numériques. Une propriété intéressante est que plus le score est haut pour un motif donné, plus la *confusion* est forte entre les avatars $a \in \mathtt{A}$ classifiés par ρ dans $\tilde{C}^{\rho}$ et ainsi plus ils sont de bons candidats pour être regroupés. Cette propriété découle directement du choix de notre opérateur $\sqcap$ en tant qu'intersections d'ensembles flous, qui se comporte comme un opérateur pessimiste (avec un retour des valeurs minimales). En revanche, on constate en regardant le treillis de la Figure 5.3 que l'ensemble des singletons ont un score de 1, étant donné que l'on additionne l'ensemble des valeurs d'une seule ligne. Ils ne nous intéressent donc pas. A partir de la matrice de confusion, nous avons les scores suivants pour les ensembles d'avatars $\{a_1, a_2\}$, $\{a_4, a_5\}$ et $\{a_1, a_2, a_4\}$.

Exemple 12 *Si nous reprenons l'exemple précédent :*

$$s(\{a_1, a_2\}^{\square}) = 0.8 \tag{5.1}$$

$$s(\{a_4, a_5\}^{\square}) = 0.75 \tag{5.2}$$

$$s(\{a_1, a_2, a_4\}^{\square}) = 0.05 \tag{5.3}$$

Si on définit un seuil $\lambda = 0.70$ à partir duquel on garde les concepts si leur score s dépasse ce seuil, alors $\{a_1, a_2\}$ et $\{a_4, a_5\}$ seront gardés et $\{a_1, a_2, a_4\}$ ne sera pas retenu, comme on le voit dans le treillis 5.3. La bordure bleue permet de séparer les ensembles dont le score est supérieur ou égal à 0.70 (en haut) par rapport à ceux dont le score est inférieur à 0.70 (en bas). Après une étape d'énumération des concepts en utilisant le seuil λ, on génère un ensemble de paires issues des concepts. Pour cela, on sélectionne les objets de l'extension et on énumère les combinaisons de paires possibles entre ces avatars. Par exemple, si on a $\{a_1, a_2, a_3\}$ alors on génère les paires (a_1, a_2),(a_1, a_3) et (a_2, a_3). Une fois cet ensemble de paires créé, nous procédons à une dernière étape qui permet de tester l'ensemble des paires selon une mesure cosinus.

Post-traitement des candidats Si nous considérons la condition de regroupement précédemment formalisée $\tilde{C}^{\rho}_{ij} \simeq \tilde{C}^{\rho}_{ji} \simeq \tilde{C}^{\rho}_{ii} \simeq \tilde{C}^{\rho}_{jj}$ et $\tilde{C}^{\rho}_{ii} + \tilde{C}^{\rho}_{ij} + \tilde{C}^{\rho}_{ji} + \tilde{C}^{\rho}_{jj} \simeq 2$. Considérons que la paire (a_i, a_j) respecte ces conditions. Il est facile de voir que (a_i, a_j) va avoir un score élevé à l'issue de l'étape précédente.

$$\begin{aligned} & \tilde{C}^{\rho}_{ij} \simeq \tilde{C}^{\rho}_{jj} \simeq min(\tilde{C}^{\rho}_{ij}, \tilde{C}^{\rho}_{jj}) \\ & \wedge \tilde{C}^{\rho}_{ii} \simeq \tilde{C}^{\rho}_{ji} \simeq min(\tilde{C}^{\rho}_{ii}, \tilde{C}^{\rho}_{ji}) \\ \implies & min(\tilde{C}^{\rho}_{ij}, \tilde{C}^{\rho}_{jj}) + min(\tilde{C}^{\rho}_{ii}, \tilde{C}^{\rho}_{ji}) \simeq 1 \end{aligned}$$

Afin de filtrer la liste des candidats de paires qui ne vérifient pas la définition du groupe d'avatar, nous proposons une mesure de similarité entre un couple de vecteurs calculés pour chaque avatar comme suit. Soit (a_i, a_j) une paire candidate, le cosinus est défini par :

Definition 62 (Cosinus de similarité)

$$cosine(\langle \tilde{C}^{\rho}_{ii}, \tilde{C}^{\rho}_{ij} \rangle, \langle \tilde{C}^{\rho}_{jj}, \tilde{C}^{\rho}_{ji} \rangle) = \frac{\tilde{C}^{\rho}_{ii}\tilde{C}^{\rho}_{jj} + \tilde{C}^{\rho}_{ij}\tilde{C}^{\rho}_{ji}}{\sqrt{(\tilde{C}^{\rho}_{ii})^2 + (\tilde{C}^{\rho}_{ij})^2}\sqrt{(\tilde{C}^{\rho}_{jj})^2 + (\tilde{C}^{\rho}_{ji})^2}}$$

Le cosinus de deux vecteurs peut être utilisé comme une mesure de similarité qui permet de déterminer si la paire candidate a de fortes probabilités d'être un groupe d'avatars. Si on considère que les traces d'un avatar a_i sont toutes correctement classifiées alors on a $\tilde{C}^{\rho}_{ii} = 1$ et si les traces de a_j sont toutes classifiées de manière incorrectes comme l'avatar a_i, ce qui signifie que $\tilde{C}^{\rho}_{ji} = 1$, alors nous avons la matrice de confusion suivante :

	a_i	a_j
a_i	1	0
a_j	1	0

Nous pouvons observer que la paire (a_i, a_j) peut être contenue dans l'ensemble des paires candidates et qu'elle va avoir un score élevé même si ce n'est pas un groupe d'avatars car elle ne vérifie pas la première condition. Le cosinus va être calculé de la manière suivante pour ce cas :

$$cosine(\langle \tilde{C}^{\rho}_{ii}, \tilde{C}^{\rho}_{ij} \rangle, \langle \tilde{C}^{\rho}_{jj}, \tilde{C}^{\rho}_{ji} \rangle) = cosine(\langle 1, 0 \rangle, \langle 0, 1 \rangle) = 0$$

ce qui signifie que la paire candidate n'est pas un groupe d'avatars intéressant. Il faut noter que la paire d'avatars telle que $a_{ii} = 1$ et $a_{jj} = 1$, le score est de 1 (soit le cosinus entre deux vecteurs parallèles) ainsi la paire n'est pas un groupe d'avatars. Un troisième cas de paires arrive quand les traces de a_i et a_j toutes classifiées de manière incorrecte comme un avatar a_k. Dans un tel cas, le score est de 0. Nous décrivons à présent l'ensemble de notre méthode avec la présentation globale de notre algorithme DEBROUILLE.

5.4 Algorithme DEBROUILLE

Notre méthode prend en paramètre une matrice de confusion normalisée générée à partie d'un ensemble de traces comportementales T. Deux seuils sont à fixer : le seuil fixant le score s minimum pour un concept et le seuil λ qui fixe la valeur minimale que doit prendre la fonction cosinus lors du post-traitement des paires. A partir de la matrice de confusion nous calculons l'ensemble des concepts possibles en utilisant l'algorithme `addIntent` [91], lors de l'étape *MineFuzzyConcepts* (Ligne 2) de l'algorithme 2. Pendant cette étape, les concepts sont élagués en fonction d'un seuil de score s. Les concepts sont triés en fonction de leur score s (Ligne 3). On parcourt ensuite chaque concept, qui sont convertis en liste de paires (Ligne 5). Par exemple, si l'extension d'un concept sont les trois avatars a_1, a_2 et a_3, nous le convertissons en paires (a_1, a_2), (a_1, a_3) et (a_2, a_3). L'ordre entre les paires du même concept est ignoré. Puis pour chaque paire du concept, si le cosinus de la paire candidate est inférieur au seuil λ et n'est pas encore présente dans la liste finale des paires P, alors celle-ci est élaguée (Ligne 7 et 8), sinon elle est

ajoutée dans la liste P. Pour finir, les paires sont triées selon leur cosinus (Ligne 9). L'algorithme `addIntent` est connu pour avoir une complexité linéaire c'est-à-dire que le nombre de concepts possibles peut croître de manière exponentielle [91] en fonction du nombre d'avatars A.

Algorithm 2 DEBROUILLE pseudo code

Require: $\tilde{C}^{\rho}$: matrice de confusion normalisée, λ seuil de la fonction cosinus, s seuil pour le score d'un concept.
Ensure: P liste des paires d'avatars triées par le cosinus
1: $P \leftarrow \emptyset$
2: $C \leftarrow MineFuzzyConcepts(\tilde{C}^{\rho}, s)$
3: $C \leftarrow$ trié C en fonction de s
4: **for all** $c \in C$ **do**
5: $\quad pairs \leftarrow$ paires issues de l'extension du concept de c
6: $\quad$ **for all** $p \in pairs$ **do**
7: $\quad\quad$ **if** $cosine(p) > \lambda$ **then**
8: $\quad\quad\quad P \leftarrow P \cup \{p\}$
9: $P \leftarrow$ trier P en fonction du cosinus
10: **return** P

5.5 Evaluation de la méthode

Pour évaluer notre méthode, nous avons choisi d'étudier un cas réel proposant de nombreuses traces comportementales. Les traces de jeu permettent de répondre à ces conditions, avec des données riches et disponibles librement en grandes quantités sur des sites de partage. Nous étudions le jeu de stratégie temps réel Starcraft 2, qui est une franchise sortie en 2010 et qui rencontre un grand succès dans le milieu du jeu de compétition (e-sport) [87]. Une partie standard confronte deux joueurs où chacun choisit une faction (Zerg, Protoss ou Terran) avec différentes faiblesses et forces (suivant le principe de pierre/feuille/ciseau). Les joueurs contrôlent des bâtiments et des unités à travers une carte en vue aérienne, collectent des ressources pour créer une armée et gagnent la partie en détruisant les forces de l'opposant. Un *joueur* a besoin d'un *avatar* pour entrer dans le système multijoueur dédié appelé `Battle.net`. Dans le contexte de l'e-sport, nous focalisons notre attention sur le problème du multi-compte, c'est-à-dire de joueurs qui utilisent plusieurs avatars. Cependant, rien n'empêche un joueur d'avoir de multiples avatars, notamment dans les tournois où chaque joueur possède un avatar pour chaque serveur *Battle.net* (USA, Europe, Korea, etc.). Une autre raison importante pour un joueur professionnel de cacher son identité est de cacher ses stratégies lorsqu'il s'entraine avant les compétitions. Pour cette raison, les cyber-athlètes peuvent créer différents avatars avec des noms tels que `lIlIllIlIlll` appelé *avatars bar code* (comme montré dans la Table 5.4) pour pratiquer sur des serveurs publics sans être reconnu. Comme on en peut le remarquer sur la figure 5.4, qui montre le classement

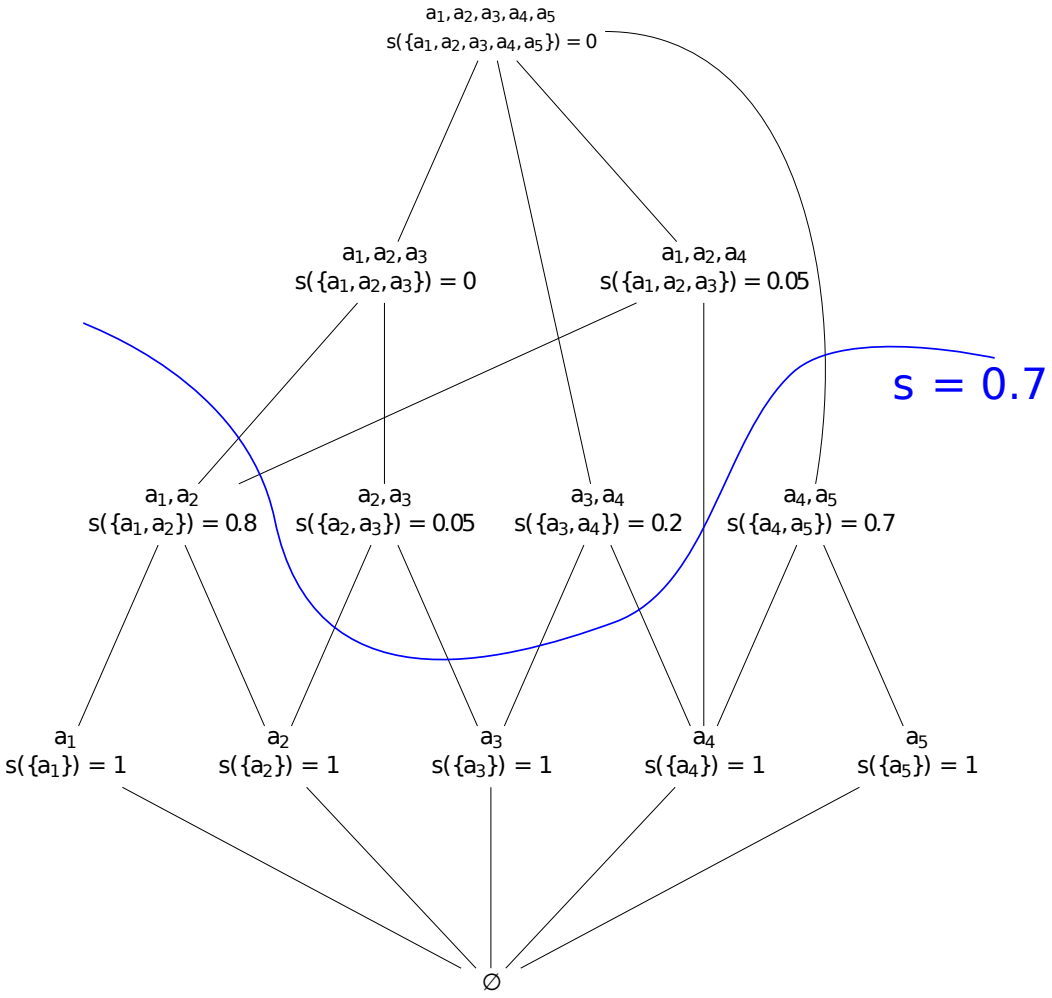


FIGURE 5.3 – Treillis des ensembles d'agents avec un seuil de score $s = 0.70$

des 16 meilleurs joueurs sur un serveur public coréen, la quasi totalité des joueurs sont anonymes. Pratiquer sur ces serveurs publics consiste en la majeure partie de leur activité [87]. Notre but est de formaliser et résoudre le problème qui consiste à trouver les avatars qui appartiennent au même joueur sans avoir aucune information sur le joueur mais seulement sur les informations de ses parties.

5.5.1 Données comportementales de jeu

Toutes les parties de Starcraft 2 sont enregistrées dans un fichier appelé *replay* qui contient toutes les données nécessaires pour que le moteur de jeu rejoue entièrement la

FIGURE 5.4 – Le top 16 des meilleurs joueurs coréens sur un serveur public

Avatar	Trace de jeu	Résultat
RorO	s,s,hotkey4a,s,hotkey3a,s,hotkey3s, ...	Lose
TAiLS	Base,hotkey1a,s,hotkey1s,s,hotkey1s, ...	Win

TABLE 5.3 – Traces pour un match entre deux joueurs

partie. Les parties sont partagées sur des sites dédiés [13]. Un ensemble d'outils d'analyse permettant d'extraire les informations de jeu sont disponibles librement [14]. Les données comportementales de Starcraft 2 de chaque joueur peuvent être modélisées sous la forme d'une série d'actions, appelés traces, comme présentées dans la Table 5.3. Les actions de base possèdent des sémantiques telles que : 'sélection d'une unité', 'attaque avec l'unité sélectionnée', 'collecter une ressource avec l'unité sélectionnée', etc. Certaines de ces actions peuvent être faites avec la souris via des menus sur l'écran. Néanmoins, pour accroitre le nombre d'actions faites dans le jeu, les joueurs experts préfèrent assigner ces actions directement à des touches de leur clavier et ainsi utiliser des *raccourcis claviers*, appelés également des *control groups* en anglais [92]. Toute action, faite à la souris ou au clavier, est stockée dans des fichiers qui sont disponibles en ligne et utilisables pour permettre d'étudier d'autres stratégies de joueurs. Les replays constituent des données

13. `http://wiki.teamliquid.net/starcraft2/Replay_Websites`
14. `http://sc2reader.readthedocs.org/`

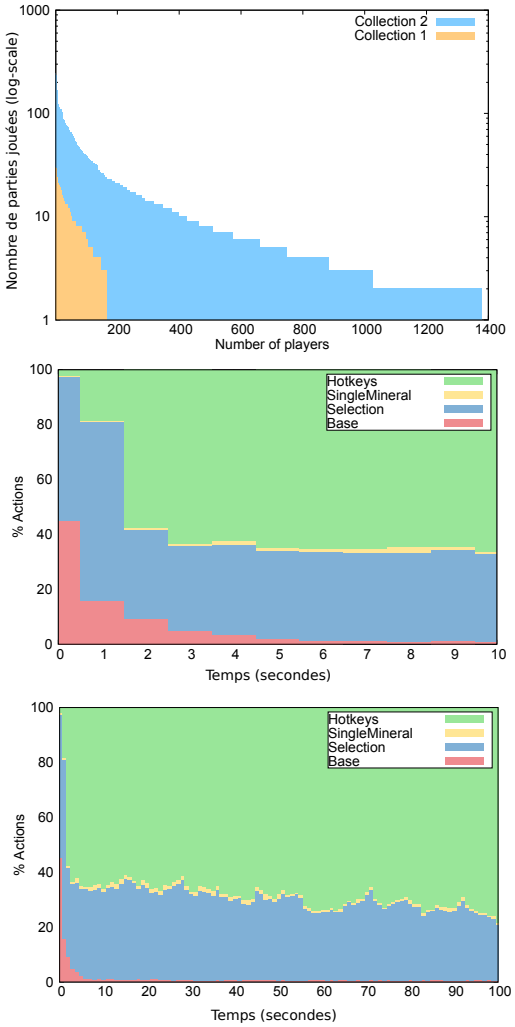


FIGURE 5.5 – Distribution du nombre de parties par avatar, proportion des actions effectuées pour les 10 (resp. 100) premières secondes.

comportementales extrêmement riches, et nous les utilisons pour répondre à notre problème. La Table 5.3 montre deux traces associées à une unique partie : Le joueur *TAiLS* commence par sélectionner avec sa souris un bâtiment appelé Base, l'assigne à la touche 1 (a), sélectionne quelques unités avec la souris (s), sélectionne des unités via la touche 2 (s), etc. En utilisant les outils d'analyse des parties, nous avons créé deux collections pour l'évaluation de notre méthode de détection d'agents usurpateurs.

COLLECTION 1 – **Parties sans doublons** Cette collection a été choisie pour étudier l'efficacité des méthodes de classification à reconnaître les avatars à partir des traces. Ainsi, nous avons sélectionné une collection de parties qui ne contiennent aucun avatar

dupliqué. C'est le cas pour les championnats du monde 2014(la seconde saison [15]) durant laquelle les utilisateurs sont obligés de s'enregistrer avec leurs vrais noms. Cette collection contient un total de 955 parties de haut niveau et 171 joueurs uniques.

Collection 2 – **Parties avec des doublons** Nous avons récupéré toutes les parties disponibles du site web *Spawning Tool*[16] le mois de Juillet 2014, pour un total de 10,108 parties et 3,805 avatars. Cette collection correspond à une situation réelle, et est utilisée pour évaluer notre approche de détection d'avatars dupliqués comme présenté dans la Section 5.5.

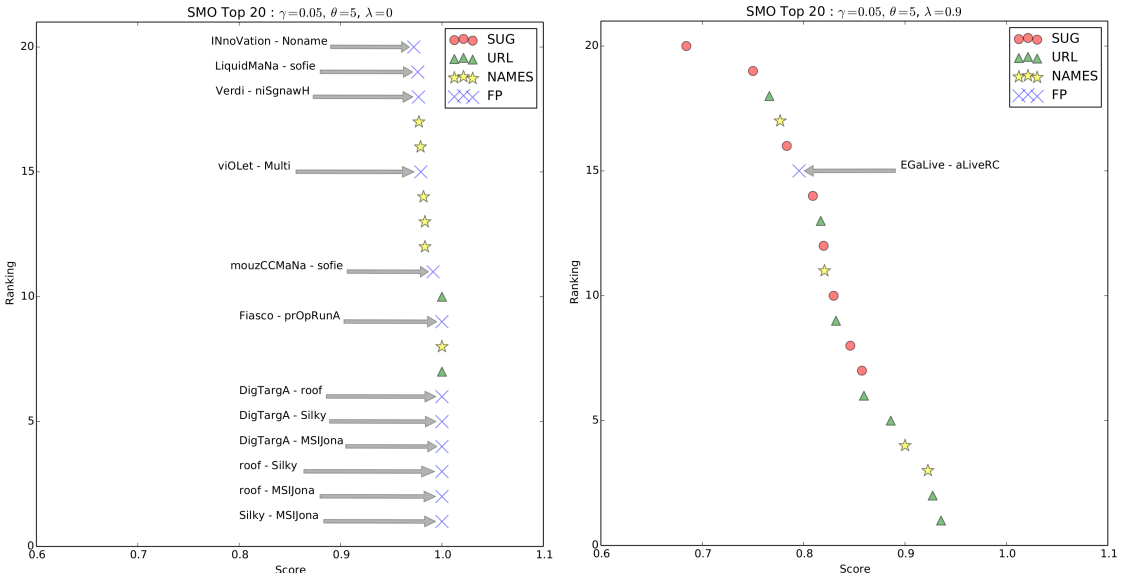


Figure 5.6 – Classement des paires candidates avec $\lambda = 0$ (gauche) et avec $\lambda = 0.9$ (droite)

La Figure 5.5 montre la distribution du nombre de parties par avatars pour les deux collections de parties. La distribution de la Collection 2 correspond à une distribution en longue traîne où 10% des joueurs participent à plus de 67% des parties. La distribution pour la Collection 1 est expliquée par le processus d'élimination des qualifications du tournoi, ce qui signifie que la distribution des parties jouées par les utilisateurs augmentent graduellement en même temps qu'ils grimpent les échelons du tournoi. En moyenne chaque joueur participe à 5 et 9 parties dans les Collections 1 et 2 respectivement. Les graphiques au centre et à droite de la Figure 5.5 illustrent la proportion de chaque type d'actions utilisées comme caractéristiques pour la classification des avatars par notre approche. Plus particulièrement, ces figures montrent les actions des 10 et 100 premières secondes du jeu, respectivement. La sélection d'objets est l'évènement le plus présent pendant les premières secondes du jeu (la phase de *warm up*), pour un total de 80% des actions, après lesquelles les raccourcis claviers deviennent les actions

15. http ://wcs.battle.net/sc2/en/articles/wcs-2014-season-2-replays
16. http ://spawningtool.com/

Account : (eu,2452136)	
Avatar	**URL**
MinChul	http://eu.battle.net/sc2/en/profile/2452136/1/MinChul/
SKMC	http://eu.battle.net/sc2/en/profile/2452136/1/SKMC/
Account : (eu,4233584)	
Avatar	**URL**
INnoVation	http://eu.battle.net/sc2/en/profile/4233584/1/INnoVation/
lIlIllIlIlll	http://eu.battle.net/sc2/en/profile/4233584/1/lIlIllIlIlll/
Account : (us,288081)	
Avatar	**URL**
Minigun	http://us.battle.net/sc2/en/profile/288081/1/Minigun/
ROOTMinigun	http://us.battle.net/sc2/en/profile/288081/1/ROOTMinigun/
Account : (us,2929052)	
Avatar	**URL**
ROOTheognis	http ://us.battle.net/sc2/en/profile/2929052/1/ROOTheognis/
Account : (us,3023756)	
Avatar	**URL**
MinChul	http ://us.battle.net/sc2/en/profile/3023756/1/MinChul/

TABLE 5.4 – Un exemple de cinq comptes Battle.net leurs avatars respectifs

les plus importantes. C'est la raison principale pour laquelle nous avons utilisés la fréquence de sélection des objets comme caractéristiques (dans un travail précédent, seuls les raccourcis claviers sont utilisés comme caractéristiques [92]). Le graphique à droite de la Figure 5.5 peut être expliqué de la manière suivante. Dans les premières minutes du jeu, comme le joueur n'a pas beaucoup d'options, il y a une forte proportion de clics et de sélections. Après 2 ou 3 minutes, l'utilisateur a accès à plus d'options pour ensuite assigner des raccourcis claviers. La majeure partie des raccourcis sont assignés une seule fois durant le jeu, ainsi la plus forte proportion de ces évènements sont faits durant les 200 premières secondes. Pour le reste du jeu, ces proportions se stabilisent.

5.5.2 Détermination de la validité d'un doublon

Dans cette section, nous exposons une procédure d'évaluation détaillée utilisée pour les expérimentations afin de définir la capacité de notre approche à trouver les doublons.

Similarité du nom des avatars Comme nous n'avons aucune information à propos des utilisateurs dissimulés derrière les avatars, il n'est pas possible d'évaluer les paires de candidats pour le regroupement en utilisant une vérité terrain. A la place, nous faisons une évaluation indirecte de notre approche en utilisant trois stratégies différentes. Dans la Figure 5.7, nous montrons le modèle de Starcraft 2 : un utilisateur possède un compte qui va contenir des avatars qui vont générer des traces en jouant des parties. La Table 5.4 montre comment ce modèle fonctionne.

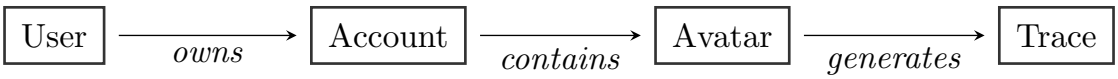


FIGURE 5.7 – Le modèle de génération de traces de Starcraft 2

Pseudonymes Chaque avatar est associé avec un pseudonyme non unique. Ces noms sont choisis par les utilisateurs et peuvent être changés à n'importe quel moment. Pour changer ce nom, les utilisateurs doivent payer une somme à l'éditeur du jeu vidéo. Ce qui signifie que si les joueurs peuvent changer leurs pseudonymes, cela reste peu fréquent. Nous pouvons identifier trois situations pour le changement de nom. Premièrement, les utilisateurs veulent être anonymes et changent leur nom pour éviter d'être reconnu par les autres joueurs. Nous considérons ce moyen d'anonymisation comme peu coûteux, puisqu'il évite de recréer un compte et ainsi devoir reclasser son compte dans une ligue (très coûteux en temps de jeu). Deuxièmement, les joueurs peuvent rejoindre une équipe, qui sont des joueurs qui jouent souvent ensemble, contre d'autres équipes. Le nom de l'équipe est accolé à celui du pseudonyme. Troisièmement, les joueurs peuvent changer leur pseudonyme pour une autre raison. Il est clair que quand on trouve deux avatars candidats pour le regroupement, nous ne pouvons pas simplement comparer leurs noms. D'une part, les noms très populaires comme 'Batman' ou 'Superman' peuvent être utilisés par de très nombreux joueurs ; d'autre part, même si le joueur a différents comptes, et qu'on les identifie pour le regroupement, rien n'empêche l'utilisateur d'utiliser différents noms d'avatars pour ces comptes. Ainsi, les noms sont des indicateurs faibles pour le regroupement d'avatars.

URL Dans Starcraft 2, tel que décrit dans la Table 5.4, un avatar est associé avec un compte unique. Ce compte peut être identifié par une URL associée à un avatar qui contient des informations à propos de sa localisation (Union Européenne, États Unis, Corée, etc.), le numéro d'ID et le nom de l'avatar. Comme nous l'avons dit, les utilisateurs sont libres de changer le nom de leurs avatars en payant. Quand ceci est fait, l'URL change en retirant l'ancien nom et en incluant le nouveau. Néanmoins, comme le compte est le même la localisation et le numéro d'ID restent les mêmes. L'URL est un indicateur fort pour le regroupement d'avatars puisque deux avatars, avec le même compte associé, correspondent au même utilisateur. Nous pouvons donc intégrer les traces de ces avatars en un seul avant l'étape de classification. Mais il est aussi utile de ne pas les regrouper, pour pouvoir créer une vérité terrain. En effet, si notre système peut regrouper des avatars de comptes différents, il peut regrouper les avatars d'un même compte.

Doublons Soit un ensemble de traces T et l'ensemble des avatars A, nous générons une partition de A qui a deux différents sous ensembles : A_γ et A_θ (une partition signifie $A = A_\gamma \cup A_\theta$ et $A_\gamma \cap A_\theta = \emptyset$). Pour chaque $a \in A_\theta$, nous générons une partition dans T_a avec les composants T_{a^1} et T_{a^2} où a^1, a^2 sont appelés des *doublons* de a. Soit $\tilde{A}_\gamma$ l'ensemble de tous les doublons, nous construisons l'ensemble $\tilde{A} = \tilde{A}_\gamma \cup A_\theta$.

Pour évaluer notre approche nous avons mesuré la précision, le rappel et la F-measure pour les 100 premiers groupes d'avatars triés par leur score. Soit ce classement r :

$$precision(r) = \frac{TP}{TP + FP}$$
$$Rappel(r) = \frac{TP}{TP + FN}$$
$$F\text{-}measure(r) = \frac{2 \cdot precision(r) \cdot Rappel(r)}{precision(r) + Rappel(r)}$$

Où TP, FP et FN sont les vrais positifs, faux positifs et faux négatifs respectivement. Nous avons considéré les vrais positifs comme les combinaisons des noms d'avatars ($NAMES$), les urls URL et les aliases (SUG). Les faux positifs sont n'importe quelle paire de candidats qui ne sont pas dans l'ensemble des vrais positifs pour un classement donné. Il est intéressant de remarquer qu'une paire considérée comme un faux positif dans cette définition peut ne pas réellement en être un. Nous les considérons comme faux positifs puisque nous n'avons pas assez d'informations pour les considérer comme des vrais positifs, cela signifie que les noms d'avatar ne correspondent pas, les URLs sont différentes et qu'ils ne font pas partie de l'ensemble des doublons générés. C'est en fait les paires que l'on recherche. Les faux négatifs sont des paires de candidats qui peuvent être considérés comme des vrais positifs, mais qui n'apparaissent pas dans le classement. Le graphique à gauche dans la Figure 5.6 montre l'ensemble initial de paires candidates extraites depuis la matrice de confusion générée par l'algorithme de classification Sequential Minimization Optimization (SMO) implémenté dans Weka. Les paramètres du classifieur sont laissés par défaut, tandis que les paramètres de notre approche pour cette figure sont $\gamma = 0.05$ (les avatars des 5% des joueurs les plus actifs sont transformés en doublons) et $\theta = 5$ (les joueurs avec moins de 5 parties sont exclus du jeu de données). Dans cette figure, un point représente une paire d'avatars. Si les avatars sont dupliqués, un point est représenté par un cercle rouge. S'ils ont le même compte, le point est représenté par un triangle vert et s'ils ont le même nom, le point est une étoile jaune. Dans tous les autres cas (faux positifs), le point est une croix bleue. Les faux positifs sont annotés avec les pseudonymes des avatars. La Figure 5.6 montre le top 20 des paires sans filtrage ($\lambda = 0$) et présente de très mauvais résultats. Seuls 8 des 20 points ne sont pas des faux positifs (40% de precision). Le graphique à la droite de la Figure 5.6 montre le top 20 avec un filtrage avec le seuil $\lambda = 0.9$ avec de très bons résultats. Seulement 1 des 20 points est un faux positif et représente une paire d'avatars qui est le joueur connu sous le nom `aLive`[17]. Il est clair que c'est un cas particulier, notre système est capable de fournir une précision de 100%, même si nous rapportons simplement 95% (19/20). Nous montrons également trois autres mesures, nommée P@10 (précision pour les 10 premiers éléments du classement), mean average precision (MAP), les courbes ROC (Receiver Operating Characteristic). Pour des descriptions détaillées de ces mesures, voir Manning et al. [67].

Dans la Figure 5.5 le graphique à la gauche montre une distribution du nombre de parties jouées par doublon. Nous supposons que les joueurs professionnels sont ceux qui sont à la tête de la courbe, c'est-à-dire ceux qui jouent le plus. Nous considérons cette hypothèse comme fiable, puisque pour devenir professionnels, les joueurs ont dû faire de nombreuses compétitions ce qui entraîne un haut taux de parties jouées. Ainsi, l'ensemble A_γ est créé à partir d'une fraction γ des avatars ayant un nombre important de parties jouées. De façon similaire, nous considérons le nombre minimum de parties (traces) que doit inclure un avatar $a \in A_\theta$. Nous partons du principe qu'un utilisateur qui a joué très peu de parties n'a pas de raison de créer un nouvel avatar différent. Cependant, nous sommes bien conscients que cela peut ne pas être le cas pour les utilisateurs qui

17. `http://wiki.teamliquid.net/starcraft2/ALive`

ont déjà un avatar différent et qui démarrent avec un second. Cela peut entraîner une distribution inégale entre les avatars et ainsi affecter les classifieurs dans leur capacité à les regrouper.

Pour un avatar avec peu de traces, le classifieur échoue à obtenir une bonne prédiction et la matrice de confusion va contenir des valeurs qui sont dues au hasard, plutôt que par le fait que deux avatars appartiennent au même utilisateur. Par exemple, si l'on considère un avatar a et que $|T_a| = 2$, sa ligne dans $\tilde{C}^\rho$ va contenir deux valeurs différentes de 0 et deux valeurs égales à 0.5. Il est facile de voir que cet avatar forme probablement un concept avec les avatars correspondant à ces colonnes. Pour éviter cela, nous utilisons un seuil θ tel que pour tout $a \in T_\theta$ on a $|T_a| \geq \theta$.

Le problème de la balance relève du fait que l'on est pas certain de la distribution du temps passé par le joueur entre ses différents avatars. Soit un utilisateur avec deux avatars, la question est de savoir s'il préfère l'un plutôt que l'autre, c'est-à-dire qu'un avatar a plus de traces qu'un autre ou s'il joue de manière égale entre les deux. C'est un point important puisque cela affecte directement l'efficacité de notre approche. Si en général, les utilisateurs jouent beaucoup plus de parties avec l'un de leurs avatars plus que les autres, le classifieur appliqué aux traces sera moins efficace et il va tendre à classifier les traces des avatars avec le moins de parties dans l'avatar avec le plus de parties. Pour étudier ce cas plus en détail, nous introduisons le paramètre β comme un coefficient de balance entre les traces distribuées des doublons. Soit un avatar a avec $|T_a| = 100$, c'est-à-dire que l'avatar possède 100 traces assignées. Quand on crée les doublons a^1 et a^2 avec $\beta = 0.5$, on obtient deux avatars dupliqués avec 50 traces associées, i.e. $|T_{a^1}| = |T_{a^2}| = 50$. Avec $\beta = 0.7$ on a $|T_{a^1}| = 70$ et $|T_{a^2}| = 30$.

5.6 Protocole expérimental

Cette section traite de l'évaluation de notre approche pour répondre au problème posé. Nous commençons par introduire les jeux de données et souligner l'efficacité de la prédiction des traces de jeu quand il n'y a pas de doublons dans les données.

Deux expérimentations principales sont menées. Avec la Collection 1, nous avons appliqué un ensemble de classifieurs pour tester l'efficacité de la prédiction d'un avatar d'une trace. Avec la Collection 2, nous appliquons une classification et un clustering pour regrouper les doublons. Les deux expérimentations partagent les étapes suivantes avec (i) la sélection de paramètres, (ii) la création des jeux de données (iii) la classification. La deuxième expérimentation possède une quatrième étape (iv) de clustering, scoring et de post-traitement.

Sélection des paramètres Tout au long de ce document, nous avons décrit un ensemble de paramètres qui permettent de compenser le fait que nous ayons très peu d'informations sur les utilisateurs que nous cherchons. Par exemple, le paramètre γ compense le fait que nous ne savons pas quelle est la proportion d'avatars dans le jeu de données qui correspond aux doublons d'un même joueur. Le paramètre β compense le fait que nous ne savons pas dans quelle proportion un avatar est utilisé. Par exemple, $\beta = 0.5$ représente le fait qu'un joueur peut utiliser ces avatars avec une proportion

égale, tandis que $\beta = 0.8$ représente le fait que les joueurs peuvent utiliser un avatar dupliqué beaucoup plus qu'un autre. Ces paramètres sont une manière de découvrir dans quelles conditions les notions de résolution des doublons interviennent. Pour cette raison, nous avons sélectionné un large spectre de combinaisons de paramètres pour nos expérimentations. La liste suivante présente les différents paramètres :

- τ : le seuil de temps pour les traces. Seul les τ premières secondes sont prises en compte dans le jeu de données pour la classification. Nous utilisons également un seuil pour le nombre des actions, ainsi seul les τ premières actions sont prises en compte
- γ : la proportion d'avatars convertis en *aliases*
- θ : le nombre minimum de parties jouées par un avatar pour qu'il soit retenu dans le jeu de données
- β : la proportion de parties attribuées pour un des deux doublons dérivés d'un avatar initial
- λ : le seuil du score cosinus

Création des bases de données Après avoir défini un ensemble de valeurs de paramètres, nous générons des jeux de données pour la classification à partir des collections 1 et 2. Chaque jeu de données contient les traces comme instances et les avatars comme labels de classes. Les caractéristiques des traces sont des vecteurs d'attributs numériques et un couple d'attributs catégoriels. La valeur de chaque dimension pour une trace donnée est le nombre de fois que l'évènement a été effectué dans la trace. Une dimension finale considère la moyenne des actions par minute associées à la trace. Les attributs catégoriels correspondent à la *race* utilisée par le joueur (les valeurs possibles sont : Protoss, Terran or Zerg) et le résultat possible de la partie (Gagné ou Perdu). Les jeux de données sont stockés dans le format attribut-relation (ARFF) issus du système Weka. Un unique jeu de donnée est construit pour chaque sélection différente d'attributs. Pour la collection 1 nous créons 92 jeux de données, tandis que pour la collection 2 nous créons 64 jeux de données.

Classification Chaque jeu de données est classifié en utilisant le logiciel d'apprentissage automatique Weka et est évalué en utilisant la validation croisée à 10 plis pour obtenir une matrice de confusion. Pour représenter la généricité de notre méthode nous utilisons quatre classifieurs : K plus proches voisins (KNN), Naive Bayes (NBAYES), J48 (J48) et Sequential Minimization Optimization (SMO). Les paramètres de ces classifieurs sont laissés par défaut.

Clustering, scoring et post-traitement Chaque matrice de confusion est traitée par l'algorithme addIntent implémenté dans Sephirot [18] pour obtenir un ensemble de concepts formels. Le scoring et le post traitement ont été implémentés via des scripts pythons.

Classification des avatars La figure 5.8 montre la précision et la AUC (Area Under the Curve) obtenue pour les 92 jeux de données créées pour la Collection 1. Le paramètre τ est échelonné exponentiellement sur 23 valeurs initialement de 10 à 90 secondes puis de 100 à 900 et finalement de 1000 à 5000 secondes (la partie la plus longue dans cette

18. https ://code.google.com/p/sephirot/

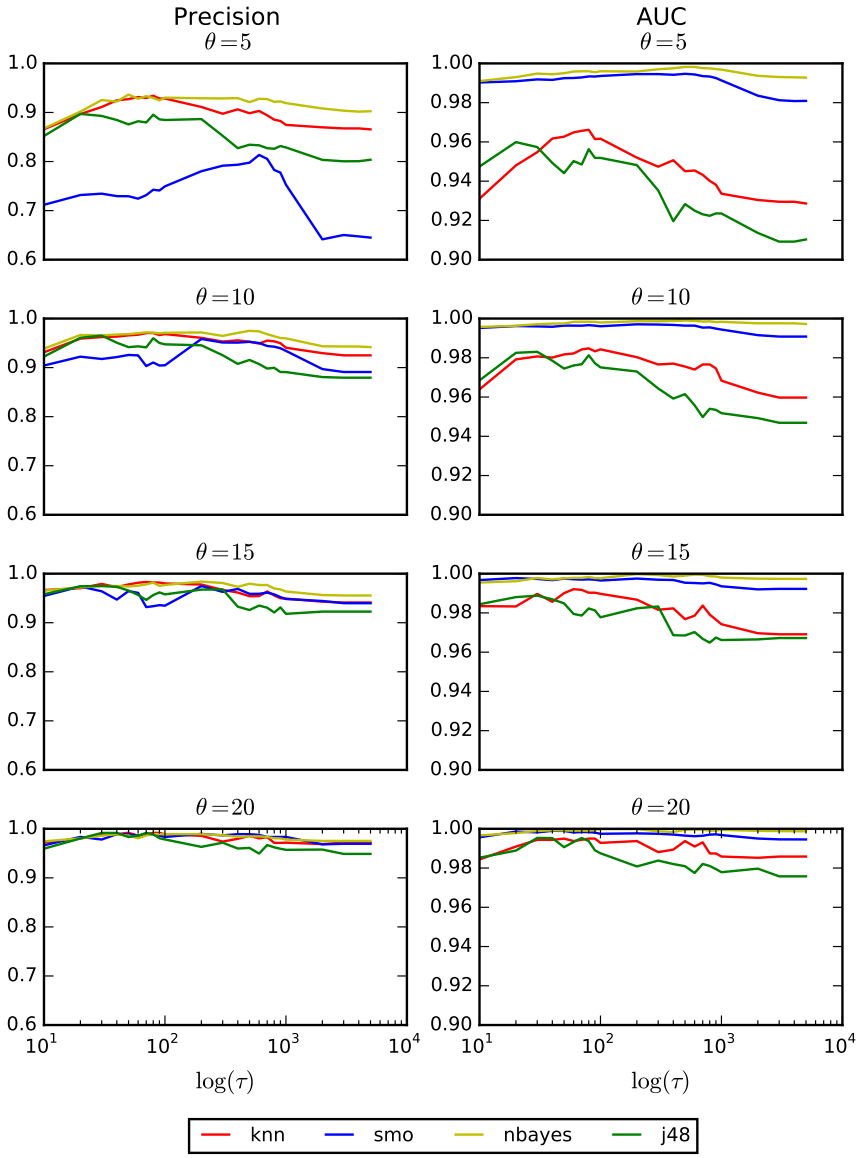


FIGURE 5.8 – Précision et distribution ROC (aire sous la courbe) de 23 points de τ pour quatre valeurs de θ avec τ variant exponentiellement.

collection dure aux alentours de 5300 secondes) et ainsi, l'axe des x de chaque figure est en échelle logarithmique. Pour chaque mesure, quatre figures correspondant à quatre différentes configurations de θ sont présentées. Chaque ligne correspond à un classifieur différent. Les figures présentent une validation empirique de notre hypothèse principale, qui supposait que les avatars sont très facilement reconnaissables en se basant sur les débuts des traces de jeu. Pour chaque différente configuration, la zone ROC est tout

le temps autour des 100% ce qui montre la robustesse de l'approche sous les différentes paramétrages. La précision se maintient tout le temps au delà de 60%, et atteint la valeur minimale avec le classifieur SMO avec $\theta = 5$ et $\tau > 1000$. Cela confirme également nos premières intuitions. Premièrement, il est difficile de reconnaître les joueurs qui ont joué très peu de parties ce qui signifie que plus la valeur du seuil θ est grande, plus le pouvoir discriminant du classifieur est grand. Deuxièmement, les joueurs sont reconnaissables dans les premières minutes du jeu. Les courbes de précisions montrent un comportement légèrement concave en fonction du seuil de temps utilisé. En outre, cela correspond avec les comportements montrés par la figure centrale de Figure 5.5. Les utilisateurs peuvent être découverts efficacement grâce aux raccourcis claviers qu'ils assignent. Au fur et à mesure que le jeu progresse, les traces peuvent différer fortement pour un même joueur, étant donné le nombre d'options qui augmente fortement en fonction des opposants.

Identification des doublons Le but de la seconde expérimentation est d'évaluer notre approche de découverte de doublons. Nous avons généré 48 jeux de données différents suivant plusieurs paramétrages. Comme nous l'avons mentionné dans l'expérimentation précédente, l'efficacité des classifieurs atteint son maximum durant les premières minutes du jeu. Ainsi, nous avons sélectionné trois différentes valeurs de τ, soit $30, 60$ et 90 secondes. Nous avons pris les mêmes valeurs que l'expérimentation précédente pour θ. Les doublons sont générés pour les premiers 5, 10, 15 et 20% des joueurs les plus actifs du jeu de données (γ). Pour cette expérimentation, nous avons une valeur de balance $\beta = 0.5$. L'ensemble des 48 jeux de données sont traités avec l'ensemble des 4 classifieurs précédemment nommés, ce qui nous donne un total de 192 matrices de confusion. La section supérieure de la Table 5.5 montre un résumé des résultats de l'évaluation utilisant les 100 premières paires d'avatars trouvées $\tau = 90$, $\theta = 20$ (248 avatars incluant les doublons), $\gamma = 0.2$ (41 doublons) et $\lambda = 0.9$. Les trois différentes parties de la table correspondent aux évaluations quand on recherche les doublons uniquement, les doublons et leurs URLs, et enfin les doublons, les URLs et les noms. Les résultats indiquent que notre approche est très efficace pour identifier les doublons sous ce paramétrage. Ceci est particulièrement vrai pour les classifieurs KNN et J48 qui donnent des valeurs de rappels très hautes. Dans la section du haut de la table, tandis que la précision est faible, on peut noter que dans le top 100, il y a seulement 41 doublons ce qui signifie que le maximum de précision est de 0.41. Le classifieur KNN est particulièrement bon dans la mesure où il possède une valeur presque parfaite (0.4 pour 0.41). L'ensemble des quatre classifieurs possèdent des hautes précisions concernant les 10 premiers résultats (P@10) avec deux ayant des scores parfaits. L'une des principales caractéristiques de notre approche est le bon classement des paires d'avatar qu'elle construit. Ce fait est confirmé par les bonnes valeurs MAP et ROC sous la courbe (AUC) que l'on obtient pour l'ensemble des 4 classifieurs. Chacune de ces mesures se dégrade quand on inclut l'ensemble des vrais positifs, les URLs et les noms. Dans le dernier cas, cela peut être compris car tous les avatars avec le même nom n'appartiennent pas nécessairement au même joueur, comme nous l'avons vu précédemment. Ainsi, les paires d'avatars avec le même nom seront distribuées plus uniformément dans le classement ou peuvent être trouvées en bas ce qui indique qu'ils ne font pas partie du même joueur. Ce fait est

visible dans l'écart entre la grande hausse de précision et la lente dégradation du rappel, c'est-à-dire que les avatars avec le même nom sont uniformément répartis entre les paires retrouvées. Comme nous l'avons dit également, les avatars avec la même URL appartiennent nécessairement au même utilisateur. Par conséquent, nous nous attendions à ce que dans les 10 premières paires retrouvées nous ayons une distribution uniforme des aliases et des URLs. Au lieu de cela, pour tous les classifieurs, P@10 possède plus de 80% aliases (tandis que le reste est toujours des URLs). La raison derrière cela est que nous avons sélectionné une balance de 0.5 pour la distribution des doublons pour les traces, alors que nous n'avons pas de contrôle pour cette valeur avec les paires d'URLs. La partie basse de la Table 5.5 montre un résumé des résultats quand on étudie uniquement les doubles et la variation de leur distribution entre les traces. Nous pouvons clairement observer que la performance de l'approche se dégrade rapidement au fur et à mesure que la distribution devient déséquilibrée (la valeur β augmente). Actuellement, pour certains classifieurs, il n'est pas possible d'obtenir de bons résultats même si nous avons mis le seuil λ à 0.8. Comme les URLs ne sont pas nécessairement équilibrées, les classifieurs tendent à prédire le label d'une trace d'un avatar avec le moins de traces avec un autre qui a le plus de traces. Les solutions pour prendre en compte les jeux de données déséquilibrés sont développées dans He et al. [43] et doivent être considérées quand on sélectionne un classifieur spécifique pour notre application.

5.7 Conclusion

Dans cette section, nous avons introduit le problème de l'identification des agents qui usurpent une identité, que nous avons appelés doublons, quand il n'existe aucune indication entre les individus et leurs avatars. C'est un problème important pour la découverte d'agents logistiques fraudeurs qui réapparaissent dans un réseau manufacturier. Notre méthode montre le fait que les données comportementales cachées d'un individu possèdent des caractéristiques spécifiques, qui permettent d'avoir des approches prédictives très précises. En revanche, ces bonnes performances se dégradent rapidement quand les données possèdent des doublons, c'est pourquoi nous nous sommes basés sur l'analyse des matrices de confusion. Ces résultats sont encourageants et permettent d'entrevoir de multiples évolutions comme perspectives. Premièrement, nous avons fait l'hypothèse que la relation entre les joueurs et les avatars étaient *une-à-plusieurs*. Si cette hypothèse est raisonnable dans le cas des joueurs experts de Starcraft 2, un modèle général suggère de proposer une relation *plusieurs-à-plusieurs*. Deuxièmement, nous nous sommes focalisés sur les joueurs experts et il est alors possible que les modèles prédictifs soient moins puissants pour les joueurs débutants, puisqu'ils utilisent les raccourcis claviers dans de moindres proportions. Ainsi, de nouvelles caractéristiques devront être proposées. Par exemple, on pourrait considérer des caractéristiques séquentielles comme dans Veloso et al. [22]. De nouvelles expérimentations ont été faites et une nouvelle méthode pour la découverte d'usurpateurs a été proposée dans Labernia et al. [55].

Paramètres : $\gamma = 0.2, \theta = 20, \lambda = 0.9, \tau = 90$						
Doublons						
classifieur	F1	MAP	Rappel	AUC	Précision	P@10
j48	0.468	0.824	0.805	0.904	0.33	1.0
naivebayes	0.226	0.740	0.390	0.915	0.16	0.8
smo	0.312	0.971	0.536	0.993	0.22	1.0
knn	0.567	0.822	0.976	0.882	0.4	0.9
Doublons & URLS						
classifieur	F1	MAP	Rappel	AUC	Précision	P@10
j48	0.588	0.907	0.606	0.866	0.57	1.0
naivebayes	0.443	0.857	0.457	0.864	0.43	1.0
smo	0.257	0.912	0.266	0.945	0.25	1.0
knn	0.670	0.937	0.691	0.874	0.65	1.0
Doublons & URLS & Noms						
classifieur	F1	MAP	Rappel	AUC	Précision	P@10
j48	0.689	0.983	0.606	0.935	0.8	1.0
naivebayes	0.560	0.943	0.492	0.906	0.65	1.0
smo	0.258	0.949	0.227	0.960	0.3	1.0
knn	0.758	0.967	0.667	0.792	0.88	1.0
Paramètres : $\gamma = 0.2, \theta = 20, \lambda = 0.8, \tau = 90$						
J48						
Balance	F1	MAP	Rappel	AUC	Précision	P@10
$\beta = 0.5$	0.925	0.996	0.929	0.955	0.920	1.0
$\beta = 0.6$	0.545	0.927	0.632	0.921	0.480	1.0
$\beta = 0.7$	0.053	0.695	0.077	0.977	0.040	0.3
Naive Bayes						
Balance	F1	MAP	Rappel	AUC	Précision	P@10
$\beta = 0.5$	0.472	0.902	0.475	0.953	0.470	0.9
$\beta = 0.6$	0.273	0.923	0.316	0.973	0.240	1.0
$\beta = 0.7$	0.197	0.9	0.288	0.978	0.150	0.9
$\beta = 0.8$	0.048	0.533	0.120	0.983	0.030	0.3
SMO						
Balance	F1	MAP	Rappel	AUC	Précision	P@10
$\beta = 0.5$	0.392	0.983	0.394	0.992	0.390	1.0
KNN						
Balance	F1	MAP	Rappel	AUC	Précision	P@10
$\beta = 0.5$	0.905	0.964	0.909	0.732	0.9	1.0
$\beta = 0.6$	0.750	0.957	0.868	0.929	0.660	1.0
$\beta = 0.7$	0.184	0.706	0.269	0.949	0.140	0.7

TABLE 5.5 – Résumé des mesures d'évaluation pour les listes de groupes d'avatars obtenues par notre approche de découverte de doublons (en haut), et le regroupement d'avatars quand on varie la balance (β) (en bas). Chaque entrée représente une matrice de confusion obtenue par le classifieur correspondant

Chapitre 6

Conclusion et perspectives

6.1 Résumé

La gestion des traces unitaires de produits captées dans le contexte manufacturier est un domaine où des problématiques spécifiques sont présentes, comme l'analyse de détournements de produits, ou de marchés parallèles. Les travaux décrits dans l'état de l'art proposent de démontrer comment les algorithmes classiques de la littérature de la fouille de données peuvent être appliqués, si nous proposons des encodages adéquats des traces unitaires. De même, les volumes de traces unitaires imposent la mise en place d'un cadre adapté mêlant différents types d'opérations de sélection pour réduire la base de traces unitaires initiale, et construire des scénarios cohérents adaptés à l'analyse de traces. En effet, les experts du domaine manufacturier s'intéressent à certaines vues de l'ensemble de leur système manufacturier, et ainsi à des traces unitaires répondant à des critères précis (types de produits, période de temps durant lesquels les produits ont voyagé, ou le marché visé). Encoder les traces unitaires dans divers contextes de fouille nécessite un cadre méthodologique permettant de spécifier les mécanismes de transformation de la connaissance issue des traces, en propriétés calculées suivant le type de structures que l'on souhaite (ensembles, séquences, graphes, etc). D'autre part, construire des contextes de fouille grâce à des processus de sélection et de transformation ouvre la voie à la proposition de méthodes spécifiques d'analyse de ces contextes. Les experts font face à des comportements anormaux observés dans les traces comme des détournements et vols de produits par rapport à un comportement normal attendus (et fixé par contrat par exemple). Cette connaissance des comportements attendus par les experts doit être étudiée en tant que telle, et il sera judicieux de proposer des méthodes d'analyse de cette connaissance pour analyser dans un premier temps les processus fréquents voulus. De plus, cette connaissance ouvre des perspectives quand on s'attache à l'analyse des traces unitaires captées. En effet, les données captées ayant des comportements attendus dévient alors de la connaissance experte. Il devient alors important de proposer des méthodes spécifiques de fouille de traces unitaires transformées utilisant activement la connaissance experte pour classer ces traces comme normales ou anormales. Or, connaître les collections de traces anormales n'est pas suffisant. En effet, il est né-

cessaire de proposer des hypothèses, sous forme d'alertes, en vue de localiser dans le temps, dans l'espace, ou tout autre contexte, les causes supposées de ces comportements anormaux.

Cependant, un autre problème important se pose, puisque rien n'empêche un agent logistique de réapparaître dans le système manufacturier sous une autre identité. Les traces unitaires, transformées en traces comportementales, peuvent également permettre de s'attaquer à ce problème. Il est naturel de se poser la question de la détection de l'identité d'un agent à partir de ces traces comportementales. Or, dans le cas d'une usurpation d'identité alors logiquement le modèle prédictif se trompera lors de la détection des agents. Il est alors intéressant d'utiliser cette 'hésitation' comme connaissance nécessaire à la découverte de ces agents possédant plusieurs identités.

Ce manuscrit comporte quatre contributions : la proposition d'un cadre méthodologique pour la fouille de traces unitaires, la recherche de motifs discriminants dans les traces unitaires à l'aide d'un modèle de filière, la découverte d'agents ayant plusieurs identités grâce à leur trace comportementale, et enfin la mise en place d'une plateforme de fouille de traces unitaires pour les experts industriels.

Le cadre méthodologique permet de prendre en charge une collection de traces unitaires sous la forme de séquences d'évènements. Nous donnons la modélisation globale de toutes les étapes : sélection, interrogation, extraction de connaissance à partir des données, et interprétation. Nous présentons particulièrement le domaine des ensembles de propriétés booléennes, et le processus de transformation des collections de traces unitaires en contextes booléens. Nous montrons comment ces contextes peuvent exprimer des scénarios variés, et comment les solveurs de fouille de données existants peuvent exprimer des connaissances utiles pour l'analyse de traces (évènements fréquents).

Au sein de ce cadre, nous proposons une méthode de découverte de motifs discriminants exprimant des hypothèses sur les contextes dérivant le plus des traces unitaires considérées comme anormales au regard d'un modèle de filière. Nous démontrons que le modèle de filière, exprimant les transitions attendues entre les différents sites (une relation commerciale pour acheminer les produits entre deux sites par exemple), est utile pour classer les traces unitaires, qui intègrent la connaissance des transitions effectuées entre les sites pour chaque produit. Cette première méthode a été appliquée sur un ensemble de données réelles issues de traces de jeu vidéo. Nous avons conçu un scénario permettant de découvrir les stratégies anormales effectuées par les joueurs ayant des comportements incohérents par rapport à la moyenne des joueurs. Les résultats démontrent la capacité d'extraire différents types d'anomalies suivant les types de propriétés générées lors de la création de contextes de fouille adaptés. Des expérimentations sur des données synthétiques ont démontrées que notre méthode passe à l'échelle en fonction de la taille du modèle de filière et/ou du nombre des traces.

Nous avons également proposé une méthode originale de découverte de paires d'agents logistiques qui sont en réalité le même agent qui aurait deux identités. Nous avons démontré son efficacité sur des traces de comportements de jeux vidéos pour détecter les joueurs jouant sous plusieurs identités différentes. Nous avons testé notre méthode selon deux scénarios, un permettant de démontrer que la découverte de l'identité d'un

agent grâce à ses traces comportementales étaient efficaces, et l'autre qui nous permet de découvrir les paires d'agents ayant la même identité.

Pour évaluer ces contributions, nous avons conçu un logiciel de fouille de traces unitaires. Il permet de construire des contextes de fouille puis d'extraire les différents motifs et d'afficher les ensembles de transitions et de traces décrites par ces motifs.

6.2 Perspectives

Nous avons évalué nos algorithmes de recherche sur des données réelles ainsi que des données synthétiques. Si le passage à l'échelle est possible concernant des volumes de traces de l'ordre de plusieurs milliers, celui ci sera beaucoup plus compliqué quand les volumes de traces atteindront le milliard, dans le cas d'applications réelles. Il serait donc intéressant de trouver des stratégies de recherche plus efficaces, que ce soit dans l'exploration des motifs que dans la parallélisation des méthodes. De plus, la transformation des traces unitaires en des structures de données autre que les ensembles peuvent permettre des perspectives intéressantes et proposer d'autres degrés de liberté aux experts. Également, l'utilisation du modèle de filière pour la classification des traces pourra être plus avancée : modèle de filière avec noeuds et/ou arêtes attribuées, autres structures qu'un graphe, utilisation de données autres que les transitions dans les traces, mise à jour du modèle à partir des motifs extraits, etc.

Les perspectives sont détaillées plus longuement dans les paragraphes suivants.

6.2.1 Contextes de fouille non booléens

Lors de la description de notre cadre méthodologique, nous avons abordé toute une série de méthodes pour transformer les traces unitaires en contexte de fouille sous la forme de données ensemblistes. Une première perspective importante est de s'appuyer sur le reste des méthodes de l'état de l'art pour proposer des méthodes de transformation, adapté aux traces, dans d'autres types de contextes : numérique, séquentiel, graphe, trajectoires spatio-temporelles. La construction de contextes de fouille non booléens permettra d'extraire d'autres types d'information, avec les solveurs adaptés. Notre méthode d'extraction de motifs intelligibles reste assez générique pour être applicable à des contextes non booléens. En effet, avant de calculer la mesure, nous appliquons un algorithme de fouille de motifs clos fréquents, il est alors envisageable d'utiliser d'autres solveurs, comme les séquences [73]. De même, le calcul du score d'anormalité se base sur les données comportementales des traces unitaires qui sont alors indépendantes du langage de motifs des descriptions, ce qui permet de classifier ces instances, quelle que soit leur structure. Une piste intéressante serait d'utiliser la généralisation de l'analyse de concepts formels, les structures de patrons, que nous avons utilisés pour notre troisième contribution, comme motifs pour caractériser les collections de traces négatives en utilisant les classes des objets sur lequel il porte (l'extension des concepts).

6.2.2 Modèles de filière augmentés

Dans ce manuscrit, nous avons décrit la connaissance experte sous la forme d'un graphe orienté, le modèle de filière. Nous nous sommes servi de la capacité d'un graphe à décrire les transitions entre les différents sites industriels, pour les comparer avec celles présentes (ou non) dans les traces unitaires. Nous avons décrit un score d'anormalité se basant sur les valeurs d'anormalité de chaque transition d'une trace. Ce score est basé sur l'absence ou pas de transitions, ou encore la longueur du chemin minimum entre deux sites (ce qui fait que la transition de deux sites consécutifs captées dans les traces est d'autant plus anormale que le chemin minimal pour relier ces sites dans le graphe est long). Notre méthode est assez générique pour permettre l'expression du concept d'anormalité entre deux sites de multiples façons, et avec des graphes orientés augmentés. Par exemple, l'ajout d'attributs sur des sites ou des transitions peut servir de base au calcul de ce score. Nous exposons dans la figure 6.1 un cas possible de traitement de données sous la forme d'intervalles attribués à chaque transition. Chaque intervalle indique dans cet exemple la durée minimale et maximale de livraison entre chaque site : $(A, C, [1, 3])$ indique que la durée de livraison entre A vers C est comprise entre 1 et 3. Comme dans notre cas d'application, il est possible de proposer aux analystes d'indiquer des valeurs manuellement, ou de déduire automatiquement à partir d'un ensemble de traces, les valeurs minimales et maximales de livraisons entre deux sites. Nous codons dans les données comportementales non pas les seules séquences de sites, mais également une valeur numérique, qui indique la durée de livraison effective (captée et calculée) entre deux sites consécutifs. Intuitivement, nous considérons que si cette valeur, pour chaque transition, est en dehors de l'intervalle, alors la transition sera anormale (également à 1). Si la transition n'existe pas, alors nous aurons également 1, comme précédemment. Ainsi, nous voyons que l'ensemble des traces, présentée dans la table en haut de la 6.1, n'est pas anormale selon le score que nous avons proposé. En effet, chaque transition est présente dans le modèle de filière. Or, si on regarde les durées de livraison captées, et qu'on les compare avec les intervalles de valeur, on remarque que la trace t_3 possède une transition $(C, G, 8)$ anormale, puisque $8 \notin [1, 3]$. A partir de cet exemple, il est possible d'imaginer d'autres structures plus avancées pour les données comportementales comme des graphes attribués ou des trajectoires spatio-temporelles.

6.2.3 Considérations algorithmiques avancées

Dans notre seconde contribution nous avons utilisé le modèle de filière pour calculer un score d'anormalité permettant de classifier les traces selon une classe binaire (normale ou anormale). On peut envisager d'utiliser une cible numérique au lieu de l'utiliser pour partitionner la base initiale de traces, et donc adapter la mesure. On peut imaginer d'utiliser ce score directement au sein de la fouille de données sous contraintes. De la même façon que la fréquence, étudier l'évolution du score d'anormalité en fonction de l'exploration des motifs, permettrait de définir une contrainte basée sur le modèle de filière. L'avantage serait de pouvoir élaguer rapidement les motifs, et de réduire significativement le temps de calcul. Il s'agira toutefois d'étudier attentivement les caractéristiques

	Données comportementales
t_1	$\langle(A,C,2),(C,D,4),(D,E,4),(E,F,5)\rangle$
t_2	$\langle(A,C,1),(C,G,4),(G,I,2),(I,J,4)\rangle$
t_3	$\langle(B,C,3),(C,G,3),(G,H,8),(H,J,3)\rangle$

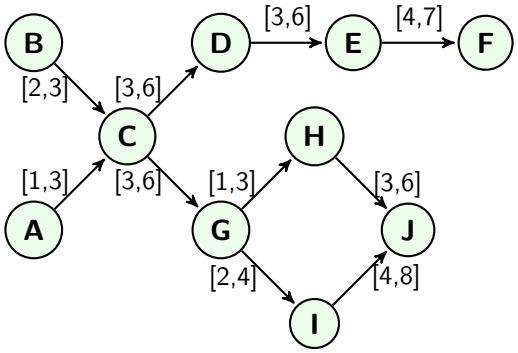


FIGURE 6.1 – Données comportementales et modèle de filière avec un attribut numérique

d'une telle contrainte pour qu'elle soit monotone ou anti-monotone (comme exposé dans Flouvat et al. [29] et Jaroszewicz [46]).

6.2.4 Impact sur le domaine manufacturier

Au sein de ce projet, la problématique de l'accessibilité des données de traçabilité est récurrente. En effet, ces données sont incomplètes et difficilement captables. Dans le cadre de certains domaines de la traçabilité, comme l'agro-alimentaire, les données sont présentes car obligatoires, cependant dans d'autres secteurs, comme les objets de luxe, la traçabilité est rarement complète. De plus, la pratique concernant le scannage des produits est très inégale : de nombreux clients scannent les produits en Chine pour obtenir des informations et très peu en Europe. Il convient donc d'établir une analyse précise des pratiques individuelles quant à l'utilisation de la traçabilité de produits. Une étude pourra être faite pour établir un système de traçabilité réduisant la perte d'informations en facilitant le scannage des produits par exemple. Nous avons décrit des cas espérés dans le cadre de cette thèse, en l'absence de données réelles de traçabilité. Avec l'évolution du système de traçabilité au sein du projet, il conviendra d'adapter les cas de fraude. Enfin, la problématique de la traçabilité en présence d'agents voulant s'assurer de la confidentialité de certaines de leurs données est encore ouverte. En effet, chaque agent possède sa propre politique de sécurité et de mise à disposition de ses données. Ainsi reconstituer les traces unitaires suppose la prise en compte de ces contraintes spécifiques à chaque agent.

Bibliographie

[1] W.M.P. Aalst. Verification of workflow nets. In *Application and Theory of Petri Nets*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer, 1997.

[2] Charu C. Aggarwal and Jiawei Han. *Managing and Mining Sensor Data*, chapter A Survey of RFID Data Processing, pages 349–382. Springer US, Boston, MA, 2013.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499, 1994.

[4] Fabrizio Angiulli and Fabio Fassetti. Exploiting domain knowledge to detect outliers. *Data Mining and Knowledge Discovery*, 28(2) :519–568, 2014.

[5] Fabrizio Angiulli, Fabio Fassetti, and Luigi Palopoli. Detecting outlying properties of exceptional objects. *ACM Trans. Database Syst.*, 34(1) :7 :1–7 :62, April 2009.

[6] Fabrizio Angiulli, Fabio Fassetti, and Luigi Palopoli. Discovering characterizations of the behavior of anomalous subpopulations. *IEEE Trans. on Knowl. and Data Eng.*, 25(6) :1280–1292, June 2013.

[7] Hiroki Arimura and Takeaki Uno. Polynomial-delay and polynomial-space algorithms for mining closed sequences, graphs, and pictures in accessible set systems, 2009.

[8] Arnaud Giacometti, Dominique H. Li, Patrick Marcel, Arnaud Soulet. 20 Years of Pattern Mining : a Bibliometric Survey. *SIGKDD Explorations*, 2014.

[9] Bojian Xu Atalay Mert İleri, M. Oğuzhan Külekci. Shortest unique substring query revisited. In *Proceedings of the 25th Annual Symposium of Combinatorial Pattern Matching*, 2014.

[10] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 429–435, 2002.

[11] Jaume Baixeries, Mehdi Kaytoue, and Amedeo Napoli. Characterizing functional dependencies in formal concept analysis with pattern structures. *Annals of Mathematics and Artificial Intelligence*, 72(1) :129–149, 2014.

[12] Jean-Francois Boulicaut and Baptiste Jeudy. *Data Mining and Knowledge Discovery Handbook*, chapter Constraint-Based Data Mining, pages 399–416. Springer US, Boston, MA, 2005.

[13] Aleksey Buzmakov, Elias Egho, Nicolas Jay, Sergei O. Kuznetsov, Amedeo Napoli, and Chedy Raïssi. On mining complex sequential data by means of FCA and pattern structures. *CoRR*, abs/1504.02255, 2015.

[14] Olivier Cavadenti, Victor Codocedo, Jean-François Boulicaut, and Mehdi Kaytoue. Identifying avatar aliases in starcraft 2. In *2nd Workshop on Machine Learning and Data Mining for Sports Analytics at ECML/PKDD 2015, Porto, Portugal*, 2015.

[15] Olivier Cavadenti, Victor Codocedo, Jean-François Boulicaut, and Mehdi Kaytoue. When cyberathletes conceal their game : Clustering confusion matrices to identify avatar aliases. In *Data Science and Advanced Analytics (DSAA), 2015. IEEE International Conference on*, pages 1–10, 2015.

[16] Olivier Cavadenti, Victor Codocedo, Mehdi Kaytoue, and Jean-François Boulicaut. Découverte de motifs intelligibles et caractéristiques d'anomalies dans les traces unitaires. In *16ème Journées Francophones Extraction et Gestion des Connaissances, EGC 2016, 18-22 Janvier 2016, Reims, France*, pages 27–38, 2016.

[17] Olivier Cavadenti, Victor Codocedo, Mehdi Kaytoue, and Jean-François Boulicaut. What did i do wrong in my moba game ? : Mining patterns discriminating deviant behaviors. In *Data Science and Advanced Analytics (DSAA), 2016. IEEE International Conference on*, 2016.

[18] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection : A survey. *ACM Computing Surveys*, 41(3) :15 :1–15 :58, July 2009.

[19] C. Chen, D. Zhang, P. S. Castro, N. Li, L. Sun, S. Li, and Z. Wang. iboat : Isolation-based online anomalous trajectory detection. *IEEE Transactions on Intelligent Transportation Systems*, 14(2) :806–818, June 2013.

[20] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. Discovering popular routes from trajectories. In *Proceedings of the IEEE 27th International Conference on Data Engineering*, ICDE '11, pages 900–911, 2011.

[21] Hong Cheng, Xifeng Yan, and Philip S. Yu. Direct discriminative pattern mining for effective classification. In *IEEE 24th International Conference on Data Engineering*, 2008.

[22] Gessé Dafé, Adriano Veloso, Mohammed Zaki, and Jr. Meira, Wagner. Learning sequential classifiers from long and noisy discrete-event sequences efficiently. *Data Mining and Knowledge Discovery*, 2014.

[23] X. H. Dang, I. Assent, R. T. Ng, A. Zimek, and E. Schubert. Discriminative features for identifying and interpreting outliers. In *2014 IEEE 30th International Conference on Data Engineering*, pages 88–99, March 2014.

[24] Xuan Hong Dang, Barbora Micenková, Ira Assent, and Raymond T. Ng. *Machine Learning and Knowledge Discovery in Databases : European Conference, ECML*

PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III, chapter Local Outlier Detection with Interpretation, pages 304–320. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[25] Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns : Discovering trends and differences. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 43–52, 1999.

[26] Lei Duan, Guanting Tang, Jian Pei, James Bailey, Akiko Campbell, and Changjie Tang. Mining outlying aspects on numeric data. *Data Mining and Knowledge Discovery*, 29(5) :1116–1151, September 2015.

[27] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, pages 1022–1029, 1993.

[28] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Advances in knowledge discovery and data mining. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *In Proceedings of the 7th International Workshop at Inductive Logic Programming*, chapter From Data Mining to Knowledge Discovery : An Overview, pages 1–34. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996.

[29] F. Flouvat, J. Sanhes, C. Pasquier, N. Selmaoui, and J-F. Boulicaut. Improving pattern discovery relevancy by deriving constraints from expert models. In *ECAI'14*, pages 327–332, August 2014.

[30] Benhard Ganter and Rudolph Wille. *Formal Concept Analysis*. Springer, Berlin, 1999.

[31] Bernhard Ganter and Sergei O. Kuznetsov. Pattern structures and their projections. In *International Conference on Conceptual Structures (ICCS)*. Springer, 2001.

[32] Arnaud Giacometti and Arnaud Soulet. Détection de données aberrantes à partir de motifs fréquents sans énumération exhaustive. In *EGC'2016*, volume 30 of *Revue de Nouvelles Technologies de l'Informatique*, pages 51–62, Reims, France, January 2016.

[33] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 330–339, 2007.

[34] H. Gonzalez, Jiawei Han, Hong Cheng, Xiaolei Li, D. Klabjan, and Tianyi Wu. Modeling massive rfid data sets : A gateway-based movement graph approach. *Knowledge and Data Engineering, IEEE Transactions on*, 22(1) :90–104, Jan 2010.

[35] H. Gonzalez, Jiawei Han, Xiaolei Li, and D. Klabjan. Warehousing and analyzing massive rfid data sets. In *Proceedings of the IEEE 22nd International Conference on Data Engineering (ICDE)*, pages 83–83, April 2006.

[36] Joachim Gudmundsson and Marc van Kreveld. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*, GIS '06, pages 35–42, New York, NY, USA, 2006. ACM.

[37] Tias Guns. *Declarative pattern mining using constraint programming.* PhD thesis, Katholieke Universiteit Leuven – Faculty of Engineering, 2015.

[38] R. Gwadera, M. Atallah, and W. Szpankowski. Reliable detection of episodes in event sequences. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 67–74, Nov 2003.

[39] Phan Nhat Hai, Dino Ienco, Pascal Poncelet, and Maguelonne Teisseire. Mining time relaxed gradual moving object clusters. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 478–481, 2012.

[40] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2) :1–12, 2000.

[41] Bernhard Haubold, Nora Pierstorff, Friedrich Möller, and Thomas Wiehe. Genome comparison without alignment using shortest unique substrings. *BMC Bioinformatics*, 6(1) :1–11, 2005.

[42] D. M. Hawkins. *Identification of Outliers.* Springer Netherlands, 1980.

[43] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, 2009.

[44] Zengyou He, Xiaofei Xu, Joshua Zhexue Huang, and Shengchun Deng. Fp-outlier : frequent pattern based outlier detection. In *Computer Science and Information Systems*, 2002.

[45] Szymon Jaroszewicz. Using interesting sequences to interactively build hidden markov models. *Data Mining and Knowledge Discovery*, 21(1) :186–220, 2010.

[46] Szymon Jaroszewicz, Tobias Scheffer, and Dan A. Simovici. Scalable pattern mining with bayesian networks as background knowledge. *Data Mining and Knowledge Discovery*, 18(1) :56–100, 2009.

[47] ShawnR. Jeffery, Gustavo Alonso, MichaelJ. Franklin, Wei Hong, and Jennifer Widom. Declarative support for sensor data cleaning. In KennethP. Fishkin, Bernt Schiele, Paddy Nixon, and Aaron Quigley, editors, *Proceedings of the 4th international conference on Pervasive Computing*, volume 3968 of *Lecture Notes in Computer Science*, pages 83–100. Springer Berlin Heidelberg, 2006.

[48] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S. Jensen, and Heng Tao Shen. Discovery of convoys in trajectory databases. *Proc. VLDB Endow.*, 1(1) :1068–1080, August 2008.

[49] Mehdi Kaytoue, Zainab Assaghir, Sergei O. Kuznetsov, and Amedeo Napoli. Embedding tolerance relations in formal concept analysis : an application in information fusion. In Jimmy Huang, Nick Koudas, Gareth J. F. Jones, Xindong Wu, Kevyn Collins-Thompson, and Aijun An, editors, *ACM Conference on Information and Knowledge Management*, Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, pages 1689–1692, Toronto, Canada, October 2010. ACM.

[50] Mehdi Kaytoue, Sergei O. Kuznetsov, Amedeo Napoli, and Sébastien Duplessis. Mining gene expression data with pattern structures in formal concept analysis. *Information Sciences*, 181(10) :1989–2001, 2011.

[51] Mehdi Kaytoue and Amedeo Napoli. Classification de données numériques par treillis de concepts et structures de patrons. In *Journées Nationales de l'Intelligence Artificielle Fondamentale*, Marseille, France, October 2009.

[52] Randy Kerber. Chimerge : Discretization of numeric attributes. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, pages 123–128. AAAI Press, 1992.

[53] Florian Kerschbaum and Nina Oertel. Privacy-preserving pattern matching for anomaly detection in rfid anti-counterfeiting. In SiddikaBerna Ors Yalcin, editor, *Radio Frequency Identification : Security and Privacy Issues*, volume 6370 of *Lecture Notes in Computer Science*, pages 124–137. Springer Berlin Heidelberg, 2010.

[54] Sergei O. Kuznetsov and Sergei Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal Of Experimental And Theorical Artificial Intelligence*, 14 :189–216, 2002.

[55] Quentin Labernia, Víctor Codocedo, Mehdi Kaytoue, and Céline Robardet. Découverte de labels dupliqués par l'exploration du treillis des classifieurs binaires. In *16ème Journées Francophones Extraction et Gestion des Connaissances, EGC 2016, 18-22 Janvier 2016, Reims, France*, pages 255–266, 2016.

[56] J. G. Lee, J. Han, X. Li, and H. Cheng. Mining discriminative patterns for classifying trajectories on road networks. *IEEE Transactions on Knowledge and Data Engineering*, 23(5) :713–726, May 2011.

[57] Artuur Leeuwenberg, Aleksey Buzmakov, Yannick Toussaint, and Amedeo Napoli. *Exploring Pattern Structures of Syntactic Trees for Relation Extraction*, pages 153–168. Springer International Publishing, Cham, 2015.

[58] M. Lehtonen, F. Michahelles, and E. Fleisch. How to detect cloned tags in a reliable way from incomplete rfid traces. In *RFID, 2009 IEEE International Conference on*, pages 257–264, 2009.

[59] Jinyan Li, Guozhu Dong, and Xingchang Zhang. Discovering jumping emerging patterns and experiments on real datasets. In *In Proceedings of the Seventeenth International Conference on Machine Learning*, pages 551–558, 2000.

[60] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm : Mining relaxed temporal moving object clusters. *Proc. VLDB Endow.*, 3(1-2) :723–734, 2010.

[61] Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xie Xing. Discovering spatio-temporal causal interactions in traffic data streams. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 1010–1018, New York, NY, USA, 2011. ACM.

[62] C. Low-Kam, A. Laurent, and M. Teisseire. Detection of sequential outliers using a variable length markov model. In *Machine Learning and Applications, 2008. ICMLA '08. Seventh International Conference on*, pages 571–576, Dec 2008.

[63] Claudio Lucchese. Dci_closed : A fast and memory efficient algorithm to mine frequent closed itemsets. In *In Proceeding of the IEEE ICDM 2004 Workshop on Frequent Itemset Mining Implementations (FIMI'04)*, 2004.

[64] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1 : Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[65] Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3) :241–258, 1997.

[66] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.*, 1(3) :259–289, 1997.

[67] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, July 2008.

[68] Elio Masciari. A framework for outlier mining in rfid data. In *Proceedings of the 11th International Database Engineering and Applications Symposium*, IDEAS '07, pages 263–267, 2007.

[69] Jean-Philippe Metivier, Alban Lepailleur, Aleksey Buzmakov, Guillaume Poezevara, Bruno Crémilleux, Sergei O. Kuznetsov, Jérémie Le Goff, Amedeo Napoli, Ronan Bureau, and Bertrand Cuissart. Discovering structural alerts for mutagenicity using stable emerging molecular patterns. *Journal of Chemical Information and Modeling*, 55(5) :925–940, 2015.

[70] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. Wherenext : a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 637–646, 2009.

[71] Petra Kralj Novak, Nada Lavrač, and Geoffrey I. Webb. Supervised descriptive rule discovery : A unifying survey of contrast set, emerging pattern and subgroup mining. *J. Mach. Learn. Res.*, 10 :377–403, June 2009.

[72] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International Conference of Database Theory*, pages 398–416, 1999.

[73] Jian Pei, Jiawei Han, Behzad Mortazavi-asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei chun Hsu. Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th International Conference on Data Engineering (ICDE)*, pages 215–224, 2001.

[74] Jian Pei, W.C.-H. Wu, and Mi-Yen Yeh. On shortest unique substring queries. In *Proceedings of the IEEE 29th International Conference on Data Engineering (ICDE)*, pages 937–948, 2013.

[75] Helen Pinto, Jiawei Han, Jian Pei, Ke Wang, Qiming Chen, and Umeshwar Dayal. Multi-dimensional sequential pattern mining. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, pages 81–88, 2001.

[76] Marc Plantevit and Bruno Crémilleux. Condensed representation of sequential patterns according to frequency-based measures. In *IDA'09*, pages 155–166, 2009.

[77] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. *SIGMOD Rec.*, 29(2) :427–438, May 2000.

[78] Jun Rao, Sangeeta Doraiswamy, Hetal Thakkar, and Latha S. Colby. A deferred cleansing method for rfid data analytics. In *Proceedings of the 32Nd International Conference on Very Large Data Bases*, VLDB '06, pages 175–186, 2006.

[79] Xiuyao Song, Mingxi Wu, Christopher Jermaine, and Sanjay Ranka. Conditional anomaly detection. *IEEE Trans. on Knowl. and Data Eng.*, 19(5) :631–645, May 2007.

[80] Thorsten Staake, Frédéric Thiesse, and Elgar Fleisch. Extending the epc network : The potential of rfid in anti-counterfeiting. In *Proceedings of the 2005 ACM Symposium on Applied Computing*, SAC '05, pages 1607–1612, New York, NY, USA, 2005. ACM.

[81] Pei Sun, Sanjay Chawla, and Bavani Arunasalam. *Mining for Outliers in Sequential Databases*, chapter 9, pages 94–105. Society for Industrial and Applied Mathematics (SIAM), 2006.

[82] Einoshin Suzuki. Discovering interesting exception rules with rule pair. In *In J. Fuernkranz (Ed.), Proceedings of the ECML/PKDD Workshop on Advances in Inductive Rule Learning*, pages 163–178, 2004.

[83] Einoshin Suzuki and Jan M. Zytkow. Unified algorithm for undirected discovery of exception rules. *International Journal of Intelligent Systems*, 20 :673–691, 2005.

[84] L. Szathmary, A. Napoli, and P. Valtchev. Towards rare itemset mining. In *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)*, volume 1, pages 305–312, Oct 2007.

[85] Laszlo Szathmary, Petko Valtchev, and Amedeo Napoli. Finding minimal rare itemsets and rare association rules. In *Knowledge Science, Engineering and Management : 4th International Conference, KSEM 2010, Belfast, Northern Ireland, UK, September 1-3, 2010. Proceedings*, pages 16–27. Springer Berlin Heidelberg, 2010.

[86] Guanting Tang, James Bailey, Jian Pei, and Guozhu Dong. Mining multidimensional contextual outliers from categorical relational data. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, SSDBM, pages 43 :1–43 :4, New York, NY, USA, 2013. ACM.

[87] T. L. Taylor. *Raising the Stakes : E-Sports and the Professionalization of Computer Gaming*. MIT Press, 2012.

[88] Kazuya Tsuruta, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Shortest unique substrings queries in optimal time. In *Proceedings of the 40th International Conference on Current Trends in Theory and Practice of Computer Science*, 2014.

[89] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura. Lcm ver.3 : Collaboration of array, bitmap and prefix tree for frequent itemset mining. In *Proceedings of the 1st International Workshop on Open Source Data Mining : Frequent Pattern Mining Implementations*, OSDM '05, pages 77–86, New York, NY, USA, 2005. ACM.

[90] W. Van der Aalst, T. Weijters, and L. Maruster. Workflow mining : discovering process models from event logs. *Knowledge and Data Engineering, IEEE Transactions on*, 16(9) :1128–1142, 2004.

[91] Dean van der Merwe, Sergei Obiedkov, and Derrick Kourie. AddIntent : A New Incremental Algorithm for Constructing Concept Lattices. In *Concept Lattices*. Springer Berlin Heidelberg, Berlin/Heidelberg, 2004.

[92] Eddie Q. Yan, Jeff Huang, and Gifford K. Cheung. Masters of control : Behavioral patterns of simultaneous unit group manipulation in starcraft 2. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI)*, 2015.

[93] Xifeng Yan, Jiawei Han, and Ramin Afshar. Clospan : Mining closed sequential patterns in large datasets. In *SDM*, pages 166–177, 2003.

[94] Show-Jane Yen, Yue-Shi Lee, Cheng-Wei Wu, and Chin-Lin Lin. An efficient algorithm for maintaining frequent closed itemsets over data stream. In *Next-Generation Applied Intelligence*, volume 5579 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009.

[95] M.J. Zaki. Scalable algorithms for association mining. *Knowledge and Data Engineering, IEEE Transactions on*, 12(3) :372–390, 2000.

[96] M.J. Zaki and C.-J. Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering*, 17(4) :462–478, April 2005.

[97] D. Zanetti, L. Fellmann, and S. Capkun. Privacy-preserving clone detection for rfid-enabled supply chains. In *RFID, 2010 IEEE International Conference on*, pages 37–44, 2010.

[98] Yu Zheng. Trajectory data mining : An overview. *ACM Trans. Intell. Syst. Technol.*, 6(3) :29 :1–29 :41, May 2015.

[99] Yu Zheng and Xing Xie. Learning travel recommendations from user-generated gps traces. *ACM Trans. Intell. Syst. Technol.*, 2(1) :1–29, 2011.

[100] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 791–800, 2009.

Résumé

Dans le contexte manufacturier, des produits sont acheminés entre différents sites avant d'être vendus à des clients finaux. Chaque site possède différentes fonctions : création, stockage, mise en vente, etc. Les données de traçabilités décrivent de manière riche (temps, position, type d'action,...) les évènements de création, acheminement, décoration, etc. des produits. Cependant, de nombreuses anomalies peuvent survenir, comme le détournement de produits ou la contrefaçon d'articles par exemple. La découverte des contextes dans lesquels surviennent ces anomalies est un objectif central pour les filières industrielles concernées.

Dans cette thèse, nous proposons un cadre méthodologique de valorisation des traces unitaires par l'utilisation de méthodes d'extraction de connaissances. Nous montrons comment la fouille de données appliquée à des traces transformées en des structures de données adéquates permet d'extraire des motifs intéressants caractéristiques de comportements fréquents. Nous démontrons que la connaissance a priori, celle des flux de produits prévus par les experts et structurée sous la forme d'un modèle de filière, est utile et efficace pour pouvoir classifier les traces unitaires comme déviantes ou non, et permettre d'extraire les contextes (fenêtre de temps, type de produits, sites suspects,...) dans lesquels surviennent ces comportements anormaux. Nous proposons de plus une méthode originale pour détecter les acteurs de la chaîne logistique (distributeurs par exemple) qui auraient usurpé une identité (faux nom). Pour cela, nous utilisons la matrice de confusion de l'étape de classification des traces de comportement pour analyser les erreurs du classifieur. L'analyse formelle de concepts (AFC) permet ensuite de déterminer si des ensembles de traces appartiennent en réalité au même acteur.

Mots-clés: trace unitaire, produits manufacturiers, découverte de connaissances, fouille de motifs, modèle expert

Abstract

In a manufacturing context, a product is moved through different placements or sites before it reaches the final customer. Each of these sites have different functions, e.g. creation, storage, retailing, etc. In this scenario, traceability data describes in a rich way the events a product undergoes in the whole supply chain (from factory to consumer) by recording temporal and spatial information as well as other important elements of description. Thus, traceability is an important mechanism that allows discovering anomalies in a supply chain, like diversion of computer equipment or counterfeits of luxury items. In this thesis, we propose a methodological framework for mining unitary traces using knowledge discovery methods. We show how the process of data mining applied to unitary traces encoded in specific data structures allows extracting interesting patterns that characterize frequent behaviors. We demonstrate that domain knowledge, that is the flow of products provided by experts and compiled in the industry model, is useful and efficient for classifying unitary traces as deviant or not. Moreover, we show how data mining techniques can be used to provide a characterization for abnormal behaviours (When and how did they occur ?). We also propose an original method for detecting identity usurpations in the supply chain based on behavioral data, e.g. distributors using fake identities or concealing them. We highlight how the knowledge discovery in databases, applied to unitary traces encoded in specific data structures (with the help of expert knowledge), allows extracting interesting patterns that characterize frequent behaviors. Finally, we detail the achievements made within this thesis with the development of a platform of traces analysis in the form of a prototype.

Keywords: unitary trace, manufacturing product, knowledge discovery, pattern mining, expert model