Nouvelles approches pour la détection de relations utiles dans les processus : application aux parcours de santé

Benjamin Dalmas

UNIVERSITÉ CLERMONT-AUVERGNE

ÉCOLE DOCTORALE
SCIENCES POUR L'INGÉNIEUR

THÈSE

présentée par

Benjamin DALMAS

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

Spécialité : INFORMATIQUE

NOUVELLES APPROCHES POUR LA DÉTECTION DE RELATIONS UTILES DANS LES PROCESSUS : APPLICATION AUX PARCOURS DE SANTÉ

Soutenue publiquement le 06 Avril 2018 devant le jury :

Rapporteurs :	
Selmin NURCAN	Maître de Conférences HDR - Université Paris Panthéon-Sorbonne
Jérôme AZÉ	Professeur des Universités - Université de Montpellier
Examinateur :	
Walid GAALOUL	Professeur des Universités - Institut Mines-Télécom
Directrice :	
Sylvie NORRE	Professeur des Universités - Université Clermont-Auvergne
Co-encadrantes :	
Michelle CHABROL	Maître de Conférences - Université Clermont-Auvergne
Sophie RODIER	Maître de Conférences - Université Clermont-Auvergne

“If necessary for years.
If necessary alone.”

Winston Churchill, 1940

Remerciements

Ces travaux ont été réalisés au sein du Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS - CNRS UMR 6158) dirigé par Farouk Toumani. Je tiens à le remercier de m'y avoir accueilli.

Cette thèse a été encadrée par les supers nanas : Sylvie Norre, Michelle Chabrol et Sophie Rodier. Merci à toutes les trois d'avoir accepté de me suivre dans cette aventure, et surtout de m'avoir laissé prendre chacune des décisions qui m'étaient importantes. Merci aussi à Michel Gourgand pour m'avoir accueilli dans son équipe et pour m'avoir apporté ses conseils quand j'en avais besoin. Merci à Jean-Christophe, Johann et Fabien pour avoir eu la patience de travailler avec moi au DIM.

Je remercie les membres du jury : Selmin Nurcan, Jerôme Aze et Walid Gaaloul, pour avoir bien voulu juger mon travail dans les différents comités de thèse dans lesquels vous avez pu prendre part, dans la lecture de ce manuscrit et dans la soutenance orale.

Merci à la team hanabi-pepouze : Nono, Damien, Benajim, Raf, Matt, Riton, BergougnouX, Keuv, David, Marina (et bien d'autres), pour m'avoir fait perdre de (trop ?) nombreuses heures à jouer, à errer dans les bureaux de tout le monde. C'est toujours plus facile de voir qu'on n'est pas seul à ramer (même si c'est bien plus sympa de voir les petits derniers continuer à ramer une fois que c'est fini pour moi). Merci aux vieux, Nathalie et Pierre-Antoine, pour m'avoir mis le pied à l'étrier quand je suis arrivé au labo et pour m'avoir donné pas mal de pistes pour mes recherches. Merci à Laetitia pour avoir partagé mon bureau, ma musique, mon désordre et mes sapins de Nöel.

Merci à Vincent Barra et Alexandre Guitton pour m'avoir permis d'enseigner dans leur école, c'est une expérience enrichissante que je souhaite à tous. Merci au gang lait-fraise d'avoir été mes étudiants durant ces années et de m'avoir fait la ola un nombre incalculable de fois en rentrant en cours. Merci d'avoir fait de ces cours un moment durant lequel j'ai aimé penser à autre chose qu'à ma thèse, même si vous étiez bruyants, même si vos blagues étaient assez fades, même si vous aviez le segfault facile. J'espère vous avoir appris autant que j'ai pu apprendre.

Un grand merci à toute l'équipe de "Retour à l'Ecole" : Lucie, Antoine, Pauline, Mathilde, Geoffrey, Juliette, Vanessa (et encore bien d'autres !), pour tous les moments passés ensemble, pour tous ces efforts, pour ces concours gagnés et perdus, pour avoir donné un autre sens à notre passion pour les sciences.

Pour finir sur une note plus sérieuse, je tiens à remercier mes parents, mes supporters de la première heure, pour m'avoir permis de réaliser ce rêve de gosse. Merci à Aurélie pour avoir partagé ma vie pendant cette thèse, d'avoir accepté les sacrifices imposés, la distance, la pression des deadlines (et j'en passe), et d'avoir malgré tout accepté d'illustrer ce manuscrit de sa plume.

Résumé

Depuis le Baby-Boom d'après guerre, la France, comme d'autres pays, est confrontée à un vieillissement de la population et à des pathologies qui deviennent de plus en plus chroniques. Ces nouveaux problèmes de santé impliquent des prises en charge plus fréquentes, plus complexes et plus transversales. Cependant, plusieurs freins liés à l'évolution de la société où à l'organisation interne du système de santé, viennent entraver le développement de prises en charge adaptées pour répondre aux nouveaux besoins. Dans un contexte de réduction des dépenses, il est nécessaire d'avoir une meilleure maîtrise des processus de santé.

L'objectif de nos travaux est de proposer un ensemble de méthodes pour une meilleure compréhension du parcours de santé de la personne âgée en Auvergne. Pour cela, une description des parcours des personnes âgées est nécessaire pour avoir cette vue d'ensemble aujourd'hui manquante. De plus, cela permettra d'identifier les intervenants, les interactions ou encore les contraintes impliquées dans les différents parcours. Les travaux présentés dans ce manuscrit s'intéressent à deux problématiques. La première consiste à mettre au point des méthodes pour modéliser rapidement et efficacement les parcours de santé. Ces modèles permettront d'analyser comment les différents segments d'une prise en charge s'enchaînent. La seconde problématique consiste à proposer des méthodes pour extraire des informations pertinentes à partir des données selon un point de vue métier prédéfini, propre à la personne qui souhaite analyser ce modèle. Ces informations permettront par exemple de détecter des segments de parcours fréquents, ou encore anormaux.

Pour répondre à ces problématiques, les méthodes que nous proposons dans ce manuscrit sont issues du process mining et du data mining. Ces disciplines ont pour objectif d'exploiter les données disponibles dans les systèmes d'information pour en extraire des connaissances pertinentes. Dans un premier temps, nous proposons une méthodologie qui se base sur le pouvoir d'expression des modèles de processus pour extraire des connaissances intéressantes. Dans un second temps, nous proposons des techniques pour construire des modèles de processus partiels, dont l'objectif est de permettre l'extraction de fragments de comportements intéressants.

Les expérimentations effectuées démontrent l'efficacité des méthodes proposées. De plus, différentes études de cas ont été menées dans divers domaines pour prouver la généricité des techniques développées.

Mots-clés : système de santé, process mining, data mining, règles séquentielles profitables, méthodologie, modèles partiels, heuristiques, HU-LPM Discovery.

Abstract

Ever since the post-World War II baby-boom, France has had to cope with an aging population and with pathologies that have become chronic. These new health problems imply more reccurent, more complex and multidisciplinary medical care. However, multiple obstacles related to the evolution of the society or to the internal organization of the healthcare system hinder the development of new and more adapted health procedures to meet new medical needs. In a context where health expanses have to be reduced, it is necessary to have a better management of health processes.

Our work aims at proposing a set of methods to gain an understanding of the elderly health path in the Auvergne region. To this end, a description of the elderly health path is necessary in order to have this missing overview. Moreover, this will enable the identification of the stakeholders, their interactions and the constraints to which they are submitted in the different health procedures. The work presented in this thesis focuses on two problems. The first one consists in developing techniques to efficiently model health paths. With these models, we will be able to analyze how the different segments of a medical care are ordered. The second problem consists in developing techniques to extract relevant information from the data, according to a predefined business point of view, specific to the user who analyzes the model. This knowledge will enable the detection of frequent or abnormal parts of a health path.

To resolve these problems, the methods we propose in this thesis are related to process mining and data mining. These disciplines aim at exploiting data available in today's information systems in order to discover useful knowledge. In a first part, we propose a methodology that relies on the expressive power of process models to extract relevant information. In a second part, we propose techniques to build local process models that represent interesting fragments of behavior.

The experiments we performed show the efficiency of the methods we propose. Moreover, we analyze data from different application domains to prove the genericity of the developed techniques.

Keywords : healthcare, process mining, data mining, high-utility sequential rules, methodology, local process models, heuristics, HU-LPM Discovery.

Table des matières

Liste des tableaux

Table des figures

Liste des algorithmes

The blind men and the elephant

John Godfrey Saxe - 1872

It was six men of Indostan,
To learning much inclined,
Who went to see the Elephant
(Though all of them were blind),
That each by observation
Might satisfy his mind.

The First approach'd the Elephant,
And happening to fall
Against his broad and sturdy side,
At once began to bawl :
"God bless me ! but the Elephant
Is very like a wall !"

The Second, feeling of the tusk,
Cried, -"Ho ! what have we here
So very round and smooth and sharp ?
To me 'tis mighty clear,
This wonder of an Elephant
Is very like a spear !"

The Third approach'd the animal,
And happening to take
The squirming trunk within his hands,
Thus boldly up and spake :
"I see," -quoth he- "the Elephant
Is very like a snake !"

The Fourth reached out an eager hand,
And felt about the knee :
"What most this wondrous beast is like
Is mighty plain," -quoth he,-
"'Tis clear enough the Elephant
Is very like a tree !"

The Fifth, who chanced to touch the ear,
Said- "E'en the blindest man
Can tell what this resembles most ;
Deny the fact who can,
This marvel of an Elephant
Is very like a fan !"

The Sixth no sooner had begun
About the beast to grope,
Then, seizing on the swinging tail
That fell within his scope,
"I see," -quoth he,- "the Elephant
Is very like a rope !"

And so these men of Indostan
Disputed loud and long,
Each in his own opinion
Exceeding stiff and strong,
Though each was partly in the right,
And all were in the wrong !

MORAL :
So, oft in theologic wars
The disputants, I ween,
Rail on in utter ignorance
Of what each other mean ;
And prate about an Elephant
Not one of them has seen !

Introduction Générale

Cette thèse a été réalisée dans le cadre du projet *SaAge* "Parcours de Santé de la personne âgée en Auvergne : des besoins de la population à l'offre de soins". Sur ce projet ont collaboré le Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS - CNRS UMR 6158) et le Centre de Recherche Clermontois en Gestion et Management (CRCGM - EA 3849), en partenariat avec l'Agence Régionale de Santé (ARS) de l'ex région Auvergne. L'objectif de ce projet est d'utiliser la complémentarité des Sciences de Gestion et des Sciences de l'Ingénieur pour proposer un ensemble de méthodes pour une meilleure compréhension du parcours de santé de la personne âgée en Auvergne. Il a été partiellement financé par l'ex région Auvergne.

Depuis le Baby-Boom d'après guerre, la France, comme d'autres pays, est confrontée à un vieillissement de la population. Les personnes vivent plus longtemps, mais cela n'implique pas forcément qu'elles vivent en meilleure santé. Nous constatons une évolution de l'état de santé de la population, et plus particulièrement chez les personnes âgées. Plusieurs des pathologies qui autrefois étaient mortelles ne le sont plus aujourd'hui. Les pathologies ont donc tendance à devenir de plus en plus chroniques. Par conséquent, les personnes sont amenées à consulter les professionnels de santé plus souvent, et de façon plus régulière. A cela s'ajoute un accroissement du niveau de dépendance mentale et motrice. Ces nouveaux problèmes de santé impliquent des prises en charge plus fréquentes, plus complexes et plus transversales.

Il faut donc favoriser la mise en place de dispositifs de prise en charge adaptés, coordonnés et efficients car c'est un enjeu sociétal et organisationnel majeur qui implique une multiplicité d'acteurs issus de domaines différents (social, sanitaire et médico-social). Cependant, plusieurs freins viennent entraver l'adaptation du système de santé aux nouvelles prises en charge. Les différentes structures de santé ont très tôt adopté une organisation en silo ; i.e., une organisation dans laquelle les ressources sont "compartimentées" par services, ou cœur de métier. Dans cette configuration, il est très difficile pour un professionnel d'avoir une vue d'ensemble sur le parcours complet d'une personne, la connaissance de la prise en charge se limitant souvent au service dans lequel le professionnel exerce. De la même manière, les systèmes d'information ont été implémentés dans les structures de santé selon la même logique en silo. Cela a pour conséquence de rendre difficile le suivi d'un patient pris en charge dans différentes structures, voire même dans différents services d'une même structure.

Dans un contexte où la prise en charge d'un patient nécessite de plus en plus la transversalité des disciplines, il faut trouver une alternative à ce cloisonnement des domaines et professions de santé.

En vue d'une meilleure prise en charge des personnes âgées sur la région Auvergne, une description formelle des parcours des personnes âgées est nécessaire pour avoir cette vue d'ensemble aujourd'hui manquante. De plus, cela permettra d'identifier les intervenants, les interactions ou encore les contraintes impliquées dans les différents parcours. Le projet SaAge s'inscrit dans cette optique. Les travaux présentés dans ce manuscrit répondent à deux objectifs principaux. Le premier objectif consiste à mettre au point des méthodes pour modéliser rapidement et efficacement les parcours de santé. Ces modèles permettront d'analyser comment les différents segments d'une prise en charge s'enchaînent. Le second objectif consiste à proposer des méthodes pour extraire des informations pertinentes à partir des données selon un point de vue métier prédéfini, propre à la personne qui souhaite analyser ce modèle. Dans cette optique, nous introduisons les concepts d'*utilité*, de *coût* et de *profit* pour illustrer cette "importance relative" que vous voulons apporter aux caractéristiques des modèles. Ces informations permettront par exemple de détecter des segments de parcours fréquents, ou encore anormaux.

Ce manuscrit est organisé comme suit. Dans le *premier chapitre*, nous faisons un état des lieux du système de santé. Nous présentons les différents enjeux autour de l'augmentation de la population âgée et la nécessité de faire évoluer les prises en charge. Ensuite, nous mettons en évidence les freins sociétaux et organisationnels du système de santé qui impactent sa capacité à mettre en place de nouvelles procédures de prises en charge. De manière générale, un parcours de santé est assimilable à un *processus*. Nous pouvons donc nous baser sur les techniques existantes de modélisation et d'analyse de processus pour répondre à nos objectifs. Dans cette perspective, nous introduisons le Process Mining et présentons en quoi cette discipline récente mais prometteuse est une opportunité à saisir. Tout en positionnant nos travaux dans cette discipline, nous définissons les problématiques auxquelles nous souhaitons répondre dans ce manuscrit.

Ensuite, le manuscrit est divisé en deux parties. Dans la *première partie*, nous nous intéressons à la modélisation des parcours de soins et à l'enchaînement des différentes étapes. Plus particulièrement, nous voulons voir dans quelle mesure l'exécution de certaines activités impacte le déroulement d'un processus ; i.e., s'il y a des *dépendances* entre certaines activités.

Après avoir présenté les possibilités offertes par les techniques de Process Mining, le *deuxième chapitre* fait un état des lieux des différentes applications qui ont été faites dans le domaine de la santé. Dans la dernière section, nous analysons dans quelle mesure les techniques existantes sont adaptées pour répondre à nos objectifs.

Dans le *troisième chapitre*, nous présentons une approche pour détecter de potentielles dépendances entre les activités d'un processus. Après avoir présenté les concepts de base utilisés en Process Mining, nous proposons la méthodologie *KITE* dont l'objectif est l'extraction de connaissances à partir de modèles de processus. Pour répondre à nos problématiques, nous proposons *IDM* et *PRISM*, deux instanciations de la méthodologie KITE.

Dans le *quatrième chapitre*, nous testons les performances des approches proposées dans cette première partie. Les différentes expérimentations sont menées sur des jeux de données issus de la littérature, des jeux de données "réels" extraits de systèmes d'informations hospitaliers dans le cadre de nos travaux, et des jeux de données générés dans l'objectif de tester la robustesse de nos approches.

Dans la *seconde partie* du manuscrit, nous proposons une alternative aux approches

présentées dans la première partie. Pour cela, nous décidons d'enrichir le pouvoir d'expression des dépendances détectées, et de limiter l'analyse des processus sur un périmètre plus restreint.

Dans le *cinquième chapitre*, nous présentons et comparons les modèles *locaux* et *déclaratifs*, deux alternatives existantes à la modélisation "globale" de processus. Contrairement à la plupart des techniques modélisant les processus de bout en bout, ces modèles sont capables de gérer les processus peu structurés.

Dans le *sixième chapitre*, nous proposons le problème *d'extraction de modèles de processus partiels profitables*, extension du problème d'extraction de modèles de processus partiels pour la prise en compte de l'*utilité* d'un modèle pour la personne qui l'analyse. Cette extension permet une "personnalisation" de la construction de modèles pour que les informations qu'ils véhiculent soient vraiment pertinentes. Nous proposons par la suite des méthodes approchées pour répondre à ce problème dans des temps raisonnables.

Dans le *septième chapitre*, nous évaluons les performances des approches proposées dans cette seconde partie sur différents jeux de données issus de la littérature, générés ou extraits dans le cadre de nos travaux. Dans un premier temps, nous montrons comment les modèles de processus partiels profitables sont capables de gérer les processus peu structurés et de donner des informations pertinentes en fonction des intérêts de l'analyste. Dans un second temps, nous testons dans quelle mesure les méthodes approchées proposées permettent de réduire les temps de calcul nécessaires à l'extraction de tels modèles. Nous évaluons aussi l'impact des caractéristiques d'un jeu de données sur l'efficacité des méthodes approchées.

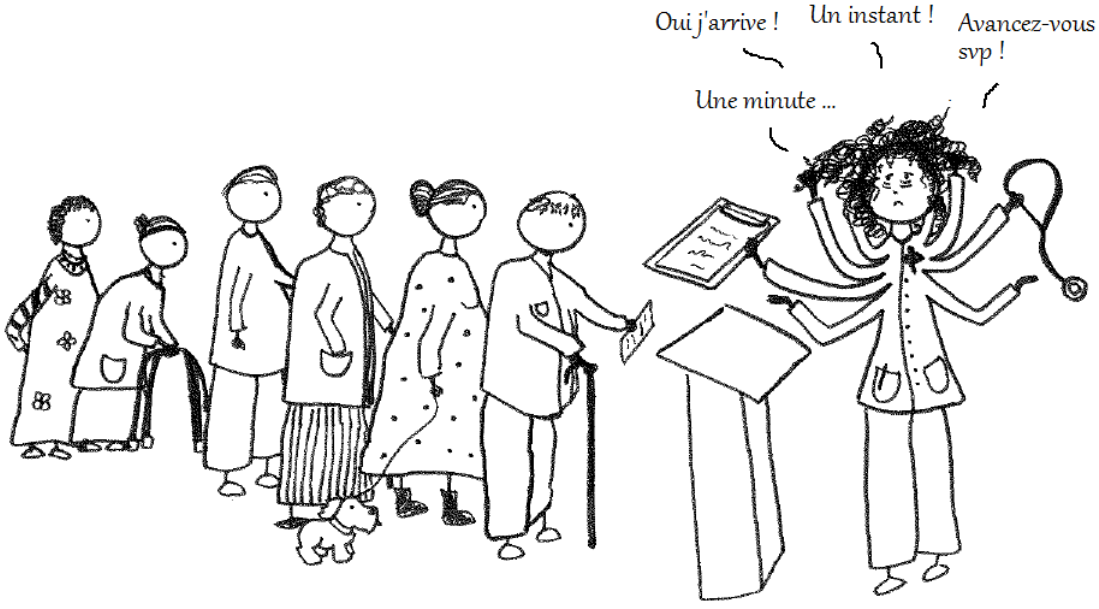
Ce manuscrit s'achève par une conclusion ainsi que par la proposition de pistes de recherche.

Bien que le domaine principal de nos travaux soit celui de la santé, nous voulons donner aux approches proposées une dimension générique, pour qu'elles soient applicables à tout domaine. Par conséquent, nous nous attachons tout au long des expérimentations à montrer cet aspect générique en analysant des processus issus de domaines d'application divers.

Chapitre 1

Contexte et Problématique

Sommaire



Introduction

De manière générale, la part que représente la population âgée croît d'année en année. En France, plus de 30% de la population aura plus de 60 ans en 2035 (Insee, 2010). En 2040, 4 personnes sur 10 habitant en zone rurale de la région Auvergne auront plus de 60 ans. Dans la même région, la part de personnes âgées incapables de réaliser les actes quotidiens devrait passer de 18% à 37% entre 2010 et 2030.

Ces prévisions impliquent que des mesures soient prises pour anticiper et éviter toute dégradation des prises en charge. Cependant, nous constatons que plusieurs freins font entrave au bon fonctionnement du système de santé français. Face à l'augmentation de la population âgée et des nouveaux besoins que cela implique, il est important de mettre en évidence les points de rupture dans le système actuel. Ce chapitre a pour objectif de faire un état des lieux du système de santé et de positionner nos travaux.

Ce chapitre est organisé comme suit. Dans la section 1.1, nous présentons le système de santé et le parcours de la personne âgée. L'objectif est de contextualiser notre périmètre d'analyse.

Dans la section 1.2, nous mettons en lumière les limites auxquelles le système de santé fait face actuellement et la nécessité de pallier ces limites.

Enfin, à partir du constat dressé dans les deux sections précédentes, nous présentons dans la section 1.3 les objectifs que nous cherchons à atteindre dans ce manuscrit ainsi que les thématiques de recherche que nous abordons pour y arriver.

1.1 Système de santé et parcours de la personne âgée

En France, les catégories "maladie" et "vieillesse" sont les principaux postes de dépenses de protection sociale. Pour subvenir à ces besoins de la population, la France déploie un arsenal d'acteurs de santé sur le territoire. Leur diversité est une force, plaçant le système de santé français parmi les meilleurs au monde. Cependant, nous verrons aussi que cette diversité est en même temps une de ses faiblesses majeures, car elle implique souvent une augmentation de la complexité dans les orientations médicales. En parallèle, la France doit faire face à un phénomène d'ampleur, celui lié à l'augmentation du nombre de personnes âgées. **Cette forte augmentation en cours est une vraie problématique pour le système de santé**. En effet, même si les dépenses de santé sont financées par des sources publiques et privées, elles sont majoritairement prises en charge par un organisme public, la Sécurité Sociale, ce qui peut présenter certaines limites lorsque la configuration de la population change.

Cette section a pour objectif de donner un aperçu du système de santé, ainsi que de la position de la personne âgée dans ce système. Après avoir présenté le système de santé en France dans la sous-section 1.1.1, nous détaillerons dans la sous-section 1.1.2 comment se caractérise l'évolution de la part de personnes âgées dans la population totale. Enfin, dans la sous-section 1.1.3, nous verrons en quoi l'augmentation de la population âgée est un défi de taille en termes de prises en charge par les structures de santé.

1.1.1 Protection sociale et système de santé

La protection sociale (ou sécurité sociale) fait référence à tous les mécanismes de prévoyance collective, permettant à un individu de faire face aux conséquences financières

des *risques sociaux*. Un "risque social" est une situation dans laquelle un individu voit sa sécurité économique compromise, par une baisse de ses revenus et/ou une hausse de ses dépenses. La maladie est un de ces grands risques sociaux, aux côtés de la vieillesse, l'invalidité ou encore le chômage. [1]

En 2013, la catégorie "maladie" était le deuxième poste de dépenses de protection sociale, avec 183,6 milliards d'euros ou 27,3% du montant total des prestations sociales (derrière la catégorie "vieillesse" avec 269,7 milliards d'euros ou 40,2% du montant total des prestations sociales). Pour prendre en charge ce risque, la France, comme beaucoup d'autres pays, s'est dotée d'un *système de santé*. L'Organisation Mondiale de la Santé (OMS) définit un système de santé comme "*l'ensemble des organisations, des institutions et des ressources dont le but est d'améliorer la santé (...)*".

Financement	**2015**
Sécurité Sociale	76,8%
État et collectivités	1,5%
Organismes complémentaires	13,3%
dont mutuelles	*7,0%*
dont sociétés d'assurance	*3,7%*
dont institutions de prévoyance	*2,6%*
Ménages	8,4%
Total	**100%**

TABLEAU 1.1 – Sources de financement de la consommation des soins et des biens médicaux en France en 2015.

Le financement du système de santé en France, comme dans tous les pays de l'OCDE, est fait aussi bien via des sources privées que des sources publiques. La part que représentent les financements publics est très variable selon les pays, même s'ils restent la principale source de financement des pays de l'OCDE, à l'exception des États-Unis. Dans le Tableau 1.1, nous présentons les différentes sources de financement en 2015 en France. Parmi les sources publiques de financement, le régime d'assurance obligatoire, représenté par la *Sécurité Sociale* et financé par les cotisations sociales représentait 76,8% du financement de la consommation des soins et des biens médicaux. L'état et les collectivités locales, au travers des impôts et taxes, assuraient 1,5% de ce financement. Parmi les sources privées de financement, les complémentaires de santé (mutuelles, sociétés d'assurances, institutions de prévoyance), organismes qui interviennent généralement pour compenser les sommes non remboursées par la Sécurité Sociale, représentaient 13,3% du financement de la consommation des soins et des biens médicaux. Enfin, les usagers du système de santé contribuent aussi directement à son financement. Les dépenses remboursées ni par la Sécurité Sociale ni par les complémentaires de santé (forfaits hospitaliers, dépassements d'honoraires) représentaient 8,4% du financement de la consommation des soins et des biens médicaux. La diversité et l'ampleur des sources de financement permettent à la France de posséder un système de santé efficace à deux niveaux : dans sa capacité d'une part à répondre aux demandes et d'autre part à couvrir les besoins de santé de la population [2]. Dans une étude publiée en 2017 qui évalue la performance d'un

1. http ://www.vie-publique.fr
2. http ://www.vie-publique.fr/decouverte-institutions/protection-sociale/risque-sante/

pays en fonction du taux de mortalité de 32 maladies pour lesquelles l'accès rapide à des soins efficaces peut améliorer le pronostic (telles que la tuberculose, le cancer du sein, la leucémie ou encore certaines maladies cardiovasculaires), la France se positionne au 15^{e} rang, sur 195 pays au total [Barber et al., 2017].

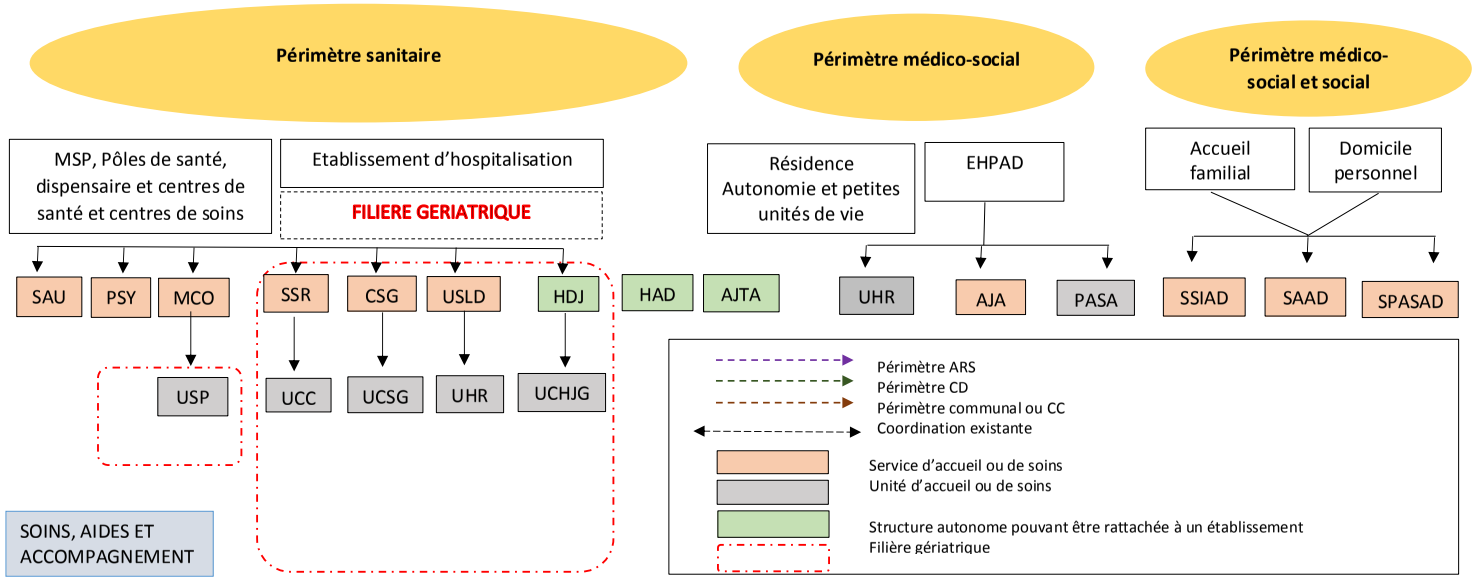
Afin de couvrir les besoins de la population en soins et biens médicaux, la France déploie sur tout le territoire un arsenal d'acteurs et de structures très divers. Nous présentons dans la Figure 1.1 une cartographie du système de santé que nous avons élaborée pour recenser toutes les parties prenantes. Dans nos travaux, nous nous intéressons plus particulièrement à une patientèle bien définie, celles des personnes âgées, dont nous détaillerons les spécificités dans la section 1.1.3. La cartographie présentée se concentre donc sur les structures qui interviennent de façon directe ou indirecte dans la prise en charge de la personne âgée. Cette cartographie n'est assurément pas exhaustive, tant le spectre des interventions est large, mais se veut représentative des principales structures rencontrées. La diversité des structures en termes de périmètres d'intervention, de statut et de sources de financement permet de les classer selon différents critères. Dans cette cartographie, nous avons délibérément choisi d'organiser les structures selon leur secteur principal d'intervention, critère sur lequel nous porterons notre attention dans la suite de ce manuscrit. Historiquement, nous comptons les secteurs du système de santé au nombre de trois, définis comme suit par [Bloch and Hénaut, 2014] :

— **sanitaire** : comprend les acteurs institutionnels, la médecine de ville et les établissements de santé essentiellement hospitaliers.

— **social** : concerne les actions visant l'insertion sociale et professionnelle des personnes.

— **medico-social** : regroupe les établissements d'hébergement accueillant des personnes en situation de handicap ou de perte d'autonomie et les services apportant de l'aide et des soins aux personnes vivant à domicile.

Il est à noter que le secteur médico-social est une spécificité française, la majorité des pays ne distinguant que les secteurs sanitaire et social. Les structures du secteur médico-social ne doivent pas être perçues comme faisant l'interface entre les secteurs sanitaire et social, mais s'orientent vers l'un ou l'autre en fonction de l'évolution des cadres juridiques. Les structures présentes dans la Figure 1.1 sont annotées par leur sigles (i.e., MCO pour Médecine-Chirurgie-Obstétrique). Nous renvoyons le lecteur vers le lexique de la cartographie pour plus de détails sur les sigles ainsi que les structures qu'ils représentent. La plupart des structures, ou services spécialisés, ont un champ d'action qui s'inscrit dans un secteur défini. Néanmoins, il arrive que certaines se trouvent à l'intersection de deux secteurs, tels que les Services de Soins Infirmiers et d'Aide à Domicile (SSIAD). En effet, il existe peu de structures qui n'interviennent que dans le secteur social. Elles proposent en général également des services de type médico-social. Nous remarquons le nombre important d'acteurs qui interviennent dans la prise en charge médicale, l'hébergement ou encore l'accompagnement de la personne âgée. Paradoxalement, cette diversité qui fait la force principale du système de santé français en lui permettant de répondre à la grande majorité des besoins de santé de la population, est à l'origine d'une de ses faiblesses principales, à savoir un partage de connaissances et une communication complexes entre des acteurs issus de secteurs différents. De plus, il est rare qu'une personne non avertie et à l'aise avec le domaine médical ait connaissance des différentes structures recensées dans la cartographie que nous avons présentée dans la Figure 1.1. Il y a donc une méconnaissance



AJTA : Accueil de jour temporaire autonome
AJA : Accueil de jour autonome
CSG : Court séjour gériatrique
HAD : Hospitalisation à domicile
HDJ : Hôpital de Jour
MCO : Médecine – Chirurgie – Obstétrique
PASA : Pôle d'activités et de soins adaptés
PSY : Psychiatrie
SAAD : Service d'aide à domicile
SAU : Service d'accueil d'urgence
SSIAD : Service de soins infirmiers à domicile
SSR : Soins de suite et de réadaptation
SPASAD : Service polyvalent d'aide et de soins à domicile
UCC : Unité cognitivo comportementale
UCHDJG : Unité de consultation et d'hospitalisation de jour gériatrique
UCSG : Unité de court séjour gériatrique
UHR : Unité d'hébergement renforcé
USLD : Unité de soins de longue durée
USP : Unité de soins palliatifs

FIGURE 1.1 – Schéma des structures du système de santé.

partagée par une majorité de la population vis-à-vis des interlocuteurs à qui elle pourrait s'adresser. Elle préfère se tourner vers les "valeurs sûres" en cas de besoin, à savoir le médecin traitant ou encore le service des urgences. Nous détaillons plus précisément les impacts de ces comportements dans la section 1.2.

1.1.2 Population âgée en France

Le concept de *personne âgée* est aujourd'hui encore assez flou, les différentes institutions de santé ne s'accordant pas sur sa définition exacte. Pour délimiter son périmètre, une idée communément admise est de définir un âge "seuil" au-dessus duquel une personne serait considérée comme âgée. Cette méthode simple trouve son intérêt notamment auprès des organismes et institutions qui sont en charge d'attribuer des prestations et dispositions relatives aux personnes âgées. En France, l'âge "seuil" initialement retenu pour déterminer si un individu est une "personne âgée" est de 60 ans, tel que défini par l'Organisation Mondiale de la Santé [3].

Cependant, la définition de ce seuil est pour beaucoup discutable. Pour prendre en compte le décalage de l'entrée dans la vieillesse au fil des générations, nous parlons aujourd'hui de "troisième âge", qui commencerait aux alentour de 60 ans ; voire même de "quatrième âge" pour les plus de 75 ans. Le glissement progressif de l'entrée dans la vieillesse a fait passer de façon plus ou moins informelle le "seuil" d'âge à 65 ans. C'est le cas par exemple pour certains barèmes d'imposition (abattement spéciaux pour personnes âgées pour l'impôt sur le revenu) ou au sein de certaines études sur les personnes âgées (telles que celles de l'Insee). Dans nos travaux nous ne dressons pas de frontières sur l'âge pour définir une personne âgée. En revanche, nous préciserons l'intervalle que nous considérons lorsque nous travaillerons sur une population âgée.

L'accroissement de la part que représente la population âgée dans la population totale est une réalité qui tend à devenir mondiale. Alors que ce phénomène n'est en 2015 d'actualité que dans les pays dits *"développés"* (Amérique du Nord, Europe Occidentale, Japon, Océanie), les pays d'Amérique du Sud, d'Asie et d'Afrique du Nord verraient leur population âgée suivre la même tendance à l'horizon 2050 [He et al., 2016]. Les pays touchés par ce phénomène aujourd'hui le seraient plus fortement encore, avec une proportion de personnes âgées représentant entre 21% et 28% de la population totale (voire même au-delà pour les pays d'Europe occidentale) contre une proportion représentant entre 14% et 20% de la population totale en 2015 [4].

Bien que le vieillissement de la population trouve certaines de ses origines dans l'allongement de l'espérance de vie dans certains pays, beaucoup s'accordent pour dire que le *"Baby Boom"* est le responsable principal [5]. Ce phénomène transitoire post-seconde guerre mondiale, s'étendant environ sur la période [1946-1974] (mais qui diffère selon les pays), a vu le nombre de naissances fortement augmenter dans les pays développés. En France, alors que la période 1910-1935 a compté environ 14 millions de naissances, la période 1950-1975 en a compté environ 21 millions, soit une augmentation de 50%. Alors que les pays ont fortement bénéficié de cette augmentation des naissances lorsque les nouveaux-nés ont participé à l'effort de la population active, ils doivent maintenant faire face à l'entrée de cette génération dans la vie inactive.

3. Organisation Mondiale de la Santé, "*Santé mentale et vieillissement*", 2016
4. US Census Bureau, 2014
5. Les conséquences du vieillissement de la population sur les dépenses de santé, CREDES, 2003

A l'horizon 2030, la cartographie des territoires français telle qu'elle est connue et définie aujourd'hui par l'Agence Régionale de Santé (ARS) risque de profondément évoluer, car ces territoires auront à affronter des défis différents selon l'ampleur du vieillissement de la population. Derrière une moyenne de 27,3% en termes de part de la population âgée dans la population totale en France, les départements vivront des réalités différentes. Par exemple, pour les projections les plus extrêmes, les départements de Seine-Saint-Denis et de la Creuse auront respectivement des prévisions de 20,5% et 45,4% en 2030 pour la part représentée par les personnes âgées. La France va donc devoir apprendre à vivre avec les vieillissements différenciés de sa population.

Dans nos travaux, nous nous intéressons particulièrement à la région Auvergne. Avec les régions du Limousin, de la Bourgogne, de la Champagne-Ardenne et de la Lorraine, elle fait partie des régions les plus "vieilles" de France, et ensemble, elles constituent ce que les géographes appellent *la diagonale du vide*. Ces régions subissent un vieillissement plus fort que la moyenne, mais qui est en partie dû à *l'héritage démographique*, c'est-à-dire le vieillissement de la population des moins de 60 ans habitant déjà la région.

Nous avons vu que l'évolution de la population âgée est une source de défis pour les territoires. Mais en plus d'être une part grandissante de la population, les personnes âgées sont aussi une patientèle particulière. Dans la sous-section suivante, après avoir défini la notion de *parcours*, nous présentons les caractéristiques spécifiques à cette patientèle.

1.1.3 Parcours de santé de la personne âgée

L'augmentation de la part représentée par la population âgée est un défi que les structures de santé vont devoir affronter. Plus de personnes âgées, de pathologies et de dépendances impliquent a fortiori plus de "parcours patient". Cependant, le concept assez vague de "parcours" mérite d'être éclairci. Dans cette sous-section, nous définissons ce concept de parcours, pour comprendre le périmètre de notre étude.

Aujourd'hui, le concept de "*Parcours de soins*" est connu sous bien des variantes, dans la littérature comme dans la vie de tous les jours, telles que "*parcours de santé*", "*trajectoire du patient*" ou encore "*chemin clinique*". Nous pourrions argumenter de la différence sémantique qu'il existe entre ces différentes notions, mais ce n'est pas l'objectif de ce manuscrit. Nous nous contentons de définir clairement les différences entre le *parcours de soins* et le *parcours de santé* pour éviter toute confusion dans la suite de ce manuscrit.

Longtemps utilisées sans distinction, les notions de parcours de soins et parcours de santé ont été comparées par [Rochette and Rodier, 2016]. En se basant sur diverses définitions données par la Haute Autorité de Santé (HAS) en 2012 ainsi que par [Benabdejlil et al., 2014], les auteurs définissent le parcours de soins comme "*l'ensemble ordonné des activités nécessaires à la prise en charge d'un individu dans une structure ciblée et pour une période délimitée allant de l'entrée de l'individu dans cette structure à sa sortie*". La Figure 1.2 présente différents segments de la prise en charge d'une personne âgée. La surveillance d'un patient en USLD ou la prise en charge d'une personne âgée par un EHPAD sont des exemples de parcours de soins.

Cependant, si nous voulons avoir un recul sur les interactions entre les différentes structures et acteurs du système de santé, le périmètre de définition du *parcours de soins* n'est pas suffisant, étant donné qu'il se concentre sur l'organisation intra-établissement ; i.e., au sein d'une structure ou d'un service de soins défini. Pour élargir le concept de parcours de soins, nous allons recourir dans ce manuscrit au concept de *parcours de santé*. Inspirées des définitions données par l'Agence Régionale de Santé (ARS) en 2012, [Rochette and

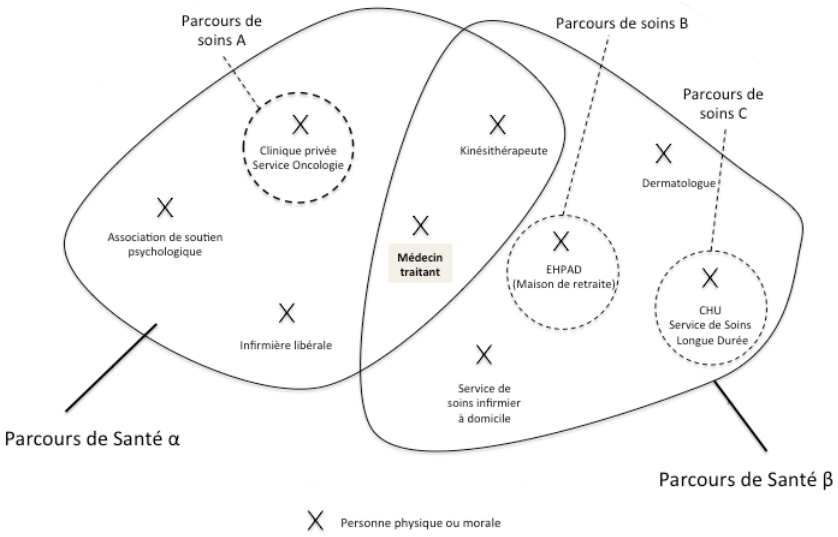


FIGURE 1.2 – Périmètre de définition du parcours de soins et parcours de santé [Rochette and Rodier, 2016].

Rodier, 2016] définissent le parcours de santé comme "*l'ensemble des étapes qu'un individu traverse dans un système sanitaire et social plus ou moins organisé dans un temps et un espace défini*". Au delà de la prise en charge de la pathologie ou de la dépendance, le parcours de santé "*va inclure les étapes en amont de prévention et les étapes en aval d'accompagnement social ou de retour à domicile*". La Figure 1.2 présente deux exemples de parcours de santé. Ils comprennent des parcours de soins, mais également les étapes préventives et d'accompagnement comme les visites chez le médecin traitant ou avec un psychologue. En plus d'une dimension impliquant les différents secteurs du système de santé, le concept de parcours va impliquer une autre dimension concernant les étapes en amont et en aval des soins directs à la personne.

L'objectif de cette section était de donner un aperçu du système de santé, ainsi que la position de la personne âgée dans ce système. Nous allons voir dans la section suivante que malgré son impact majeur sur la société, le système de santé souffre de plusieurs faiblesses. Ajouté à l'augmentation de la population âgée, plusieurs dysfonctionnements surviennent.

1.2 Un système de santé fragilisé

Bien qu'il ait prouvé son efficacité en termes de qualité de soins, le système de santé français souffre actuellement de faiblesses, qui n'ont cessé de se renforcer au cours des dernières décennies, et sur lesquelles il est important de se pencher si nous voulons en réduire les impacts négatifs. Nous verrons dans les sous-sections 1.2.1 et 1.2.2 que ces obstacles, qu'ils soient endogènes (organisation interne, interaction entre les différents acteurs, ...) ou bien exogènes (sources de financement, évolution des maladies, rapport de la population au système de santé, ...), ont des répercutions non négligeables sur la qualité et le coût des prises en charge, et par extension, sur la satisfaction de la population. Dans un contexte où l'informatique et les nouvelles technologies s'invitent partout, nous verrons dans la sous-section 1.2.3 que s'adapter à l'ère du numérique est une démarche compliquée pour le domaine de la santé et que cela peut renforcer les dysfonctionnements

déjà existants.

1.2.1 De nouveaux besoins de la population

La société, mais aussi la façon qu'a la population d'appréhender le système de santé, ont évolué au fil des années. Alors que certains de ces changements ont bénéficié à l'amélioration des prises en charge, d'autres ont renforcé les dysfonctionnements déjà existants. Aujourd'hui, la principale source d'inquiétude porte sur l'augmentation de la part que représente la population âgée dans la population totale. Mais pour bien en comprendre les enjeux, il est important de faire la distinction entre deux concepts, le *vieillissement* et la *gérontocroissance*. Le terme *vieillissement* "stricto sensu" traduit un effet de structure. Il représente, pour un territoire et une période donnés, l'augmentation du pourcentage de personnes âgées dans la population totale.

La *gérontocroissance*, quant à elle, traduit un effet de flux. Elle représente, pour un territoire et une période donnés, l'augmentation du nombre de personnes âgées. D'apparence similaires, ces deux phénomènes ont des conséquences différentes sur les territoires. Les territoires qui vont faire face à une forte gérontocroissance devront notamment s'inquiéter du nombre de places disponibles dans les structures de santé pour accueillir l'excédent migratoire de personnes âgées. Au-delà de cette problématique de dimensionnement, les territoires qui vont faire face à la problématique de vieillissement de leur population devront gérer un déséquilibre plus ou moins marqué avec la population active. Par exemple, le département de la Creuse doit s'attendre, selon les estimations, à voir sa population active devenir inférieure à celle des "60 ans et plus" en 2030. Le "*Papy Boom*" illustre parfaitement ce vieillissement de la population. Conséquence du Baby Boom, ce phénomène désigne la vague de personnes qui entre ou va entrer dans la catégorie des "personnes âgées" entre 2006 et 2034.

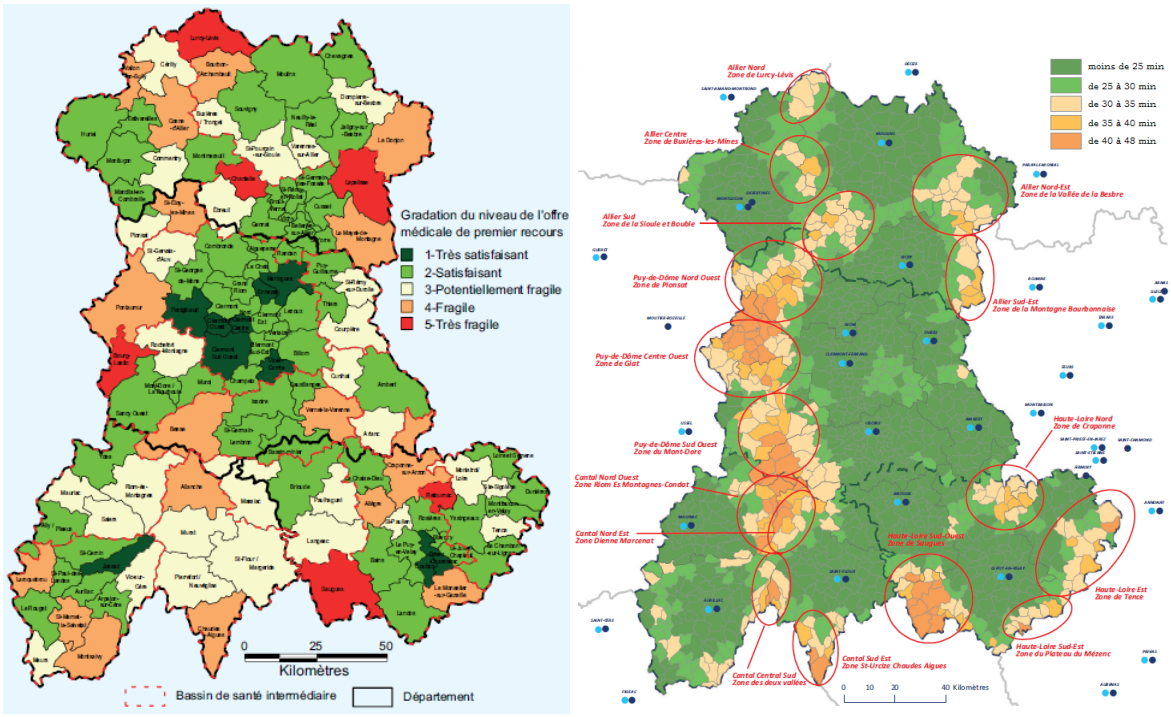
Pour anticiper l'augmentation de la population âgée, qu'elle soit due au vieillissement de la population ou à un effet de migration, une solution naïve serait de développer de nouvelles structures d'accueil et de prise en charge. Au-delà des défis que cela impliquerait en termes de financement, cette solution ne serait viable qu'à court terme. En effet, le principal défi lié à l'accroissement de la population âgée gît dans son caractère "éphémère". La réduction du taux de fécondité qui affecte les pays occidentaux depuis le milieu des années 1970 a fait de la vague de personnes âgées issues du Baby Boom un phénomène temporaire. En termes d'horizon temporel, la construction de nouveaux établissements serait donc contre-productif, étant donné qu'ils ne seraient pas effectifs avant plusieurs années (études, chantier, ...), alors que le Papy Boom a déjà commencé depuis maintenant une décennie. De plus, une fois la vague passée, il faudrait financer un nombre considérable de structures et de ressources (humaines, matérielles) que nous ne saurions mobiliser à plein temps. En résumé, les territoires vont devoir penser une solution pour prendre en charge un nombre toujours plus grand de personnes âgées, dans une période où les contraintes financières sont de plus en plus fortes, et si possible sans mobiliser de ressources supplémentaires.

A côté de la problématique de vieillissement de la population, l'évolution de la santé des personnes âgées et leur rapport au système de santé est à prendre en compte pour expliquer les dysfonctionnements actuels. C'est indéniable, les personnes vivent de plus en plus longtemps. Entre 1965 et 2015, l'espérance de vie est passée de 70,8 à 80,6 ans [6]. Il

6. *Life expectancy at birth* (Data World Bank)

faut donc s'occuper d'une personne âgée de plus en plus longtemps. Mais l'augmentation de la durée de prise en charge n'est pas le principal problème engendré par l'augmentation de l'espérance de vie. Deux phénomènes majeurs se développent et s'accroissent, la "*chronicité*" des maladies et la *pluri-pathologies*. La *chronicité* des maladies représente une tendance grandissante dans laquelle les patients sont de plus en plus sujets à développer des maladies dites "chroniques" ; i.e., une affection de longue durée qui évolue avec le temps. Plus ou moins graves selon les cas, il peut s'agir de diabète, de cardiopathies, de cancer, d'AVC, d'hypertension artérielle, de polyarthrite, d'Alzheimer ou de Parkinson, etc... Ces maladies ne tuent plus mais rendent les personnes atteintes dépendantes et changent la nature des prises en charge (suivi médical constant, soins à domicile, ...).

La chronicité des maladies s'accompagne d'un aspect *pluri-pathologique* des patients, désignant le fait qu'ils ne sont plus traités pour une seule pathologie mais pour plusieurs. Les compétences nécessaires pour traiter les différentes facettes de la pluri-pathologie se heurtent à la spécialisation des professions et des prises en charge. Il va donc de la nécessité d'avoir des parcours et prises en charge organisés et transversaux avec la mise en commun des connaissances et des ressources issues des différents secteurs du système de santé.



(a) (b)

FIGURE 1.3 – *(a)* Gradation de l'offre médicale de premiers recours en 2011 et *(b)* temps d'accès théoriques aux soins urgents en région Auvergne en 2012.

Au niveau régional, l'Auvergne se caractérise par un nombre important d'établissements et services médico-sociaux, souvent de taille modeste. Cette offre de services est globalement supérieure à la moyenne nationale, mais avec des disparités territoriales parfois importantes. Sur la Figure 1.3, nous présentons deux cartographies issues d'un rapport de l'Agence Régionale de Santé (ARS) de la région Auvergne [7]. La Figure 1.3a présente l'offre médicale sur les différentes zones de l'Auvergne et la Figure 1.3b montre les temps

7. *Portraits de santé : l'état des territoires d'Auvergne (ARS, 2013)*

théoriques moyens d'accès aux soins urgents pour les habitants de ces zones. Nous remarquons que l'accès aux soins de manière générale est inégalement réparti sur le territoire auvergnat. Il se dégage une "ceinture", partant du sud du Cantal et allant jusqu'à l'est de l'Allier, dans laquelle l'offre de soins est insuffisante et le temps d'accès aux soins urgent est considéré comme trop long. Dans le cadre des mesures annoncées par le gouvernement en 2012 en matière de santé, figurait l'accès des Français à des soins urgents à moins de 30 minutes. Cette même année, 7% de la population auvergnate se trouvait au-delà de ce seuil, ce qui est supérieur au double du ratio moyen national, qui est de 3.2%. Il y a donc une mauvaise répartition de l'offre de soins sur le territoire. Cette offre insuffisante dans cette "ceinture" coïncide avec le relief très montagneux de la région, qu'il faut prendre en compte pour répondre aux besoins de la population.

Nous avons pu constater que les structures de santé doivent composer avec une société qui évolue et des territoires inégaux dans l'accès aux soins, disparités qui impactent les différentes prises en charge. Nous allons voir dans la sous-section suivante que, bien que cette démarche soit l'occasion de concrétiser des opportunités en termes de qualité et suivi des patients, les obstacles rencontrés sont encore nombreux.

1.2.2 Un fonctionnement interne inadapté

Les études réalisées par le Haut Conseil pour l'Avenir de l'Assurance Maladie (HCAAM) en 2010 [8] ont contribué à alimenter la réflexion sur le niveau anormalement élevé des dépenses individuelles liées au "grand âge". Le principal coupable pointé du doigt est le défaut d'organisation du système de santé. Paradoxalement, c'est la structuration des acteurs par secteur qui en serait la première cause. Il existe un cloisonnement entre les différents secteurs du système de santé (sanitaire, médico-social et social) qui dégrade la qualité des prises en charge. L'OMS définit la santé comme "un état de complet bien-être physique, mental et social, et non seulement une absence de maladie ou d'infirmité" [9]. L'état social et mental est donc indissociable de l'état de santé d'une personne. Aujourd'hui, nous ne cherchons plus à soigner une pathologie, mais à prendre en charge une "condition", qui est l'agrégation d'une ou plusieurs pathologies, d'un état de dépendance ou encore d'un environnement social et familial. Cependant, en France, les différents secteurs ont évolué de façon indépendante, voire cloisonnée (méthodes d'évaluation, financement, ...). [Bloch and Hénaut, 2014] se sont penchés sur cette problématique en 2014. Les auteurs soulignent que les secteurs sanitaire et social ont développé des catégories d'intervention, des logiques professionnelles et des législations complètement différentes, ce qui rend leur coordination difficile pour prendre en compte tous les aspects de la santé tels que définis par l'OMS, et par extension rend la mise en place de politiques transversales compliquée. De plus, le cloisonnement rend les actions menées et les services fournis par chaque secteur "opaques" aux autres acteurs d'une part, et à la population d'autre part.

Il existe un manque de cohérence entre les acteurs de secteurs différents. Mais la diversité des acteurs, des statuts ou encore des cultures professionnelles a aussi dégradé le niveau de visibilité au sein d'un même secteur, d'une même structure voire au sein d'un même service. Ces problèmes sont renforcés par l'augmentation du nombre de services dans les structures, liée à la spécialisation croissante des professions médicales. Même s'il

8. *L'assurance maladie face à la crise : Éléments d'analyse* (Rapport annuel 2010, HCAAM)
9. Préambule de la Constitution de l'OMS, 1946

existe certaines coordinations, elles se développent d'une part souvent à l'intérieur d'un même service, et d'autre part de façon informelle. Face à la polyvalence que demandent les recommandations en matière de prises en charge, il est nécessaire d'améliorer l'articulation des différents secteurs du système de santé [10]. Ce manque de visibilité se répercute aussi sur les principaux intéressés, à savoir les patients. Pour les personnes non expertes du domaine, il est très difficile de connaître toutes les structures existantes et leur périmètre d'action, ayant pour conséquence un grand nombre de mauvais choix de la part des patients en terme de premier contact. La sur-représentation des personnes âgées dans les passages aux urgences est un des témoins majeurs de cette méconnaissance des structures de santé. Ajouté aux choix irrationnels en termes d'orientation médicale, dus à ce manque de visibilité, le "comportement réactif", qui désigne un phénomène de réaction plutôt que de prévention en matière de santé, renforce les mauvais choix des patients quand il s'agit de s'adresser à une structure de santé.

La maîtrise et compréhension des caractéristiques de la société impactant la qualité des soins est hors de notre domaine d'étude. Cependant, nous sommes convaincus qu'en agissant sur les aspects "organisationnels" du système de santé, il est possible de limiter, même à petite échelle, l'impact de ces évolutions sociétales et environnementales.

1.2.3 Les techniques du numérique sous-utilisées

Dans beaucoup de domaines, les changements dans la société s'accompagnent généralement d'avancées technologiques. Le domaine de la santé ne fait pas exception à la règle, la technologie faisant partie intégrante des nouvelles pratiques et procédures médicales. L'accès à des systèmes de plus en plus performants et à des connexions haut débit a permis aux acteurs et structures de santé d'être de plus en plus connectés. L'information circule plus vite et touche beaucoup plus de monde qu'avant, par exemple grâce à la mise en place de différents réseaux ou encore l'utilisation de messageries internes dans les structures. Avec le temps, les appareils et machines se complexifient et les différents actes médicaux s'appuient de plus en plus sur des outils informatisés.

Du passage d'un patient dans un service aux diagnostics et actes effectués, en passant par la simple ordonnance chez un médecin de ville, aujourd'hui en matière de parcours de santé, tout ou presque est enregistré. Disponibles dans les Systèmes d'Information Hospitaliers (SIH) dont sont équipées la plupart des structures de santé aujourd'hui, il est possible d'avoir accès à quel acte a été effectué, à quelle heure de quel jour, par quelle personne et à quel endroit. Cette nouvelle approche du suivi patient, qui engendre la génération et le stockage d'une quantité de données toujours grandissante, témoigne d'une volonté majeure d'intégrer le domaine de la santé dans l'ère du numérique. Cependant, le constat reste mitigé lorsqu'il s'agit de l'exploitation de ces données.

Alors qu'elle aurait pu être le pont qui rapprocherait les différents acteurs et structures, l'informatisation massive du système de santé a renforcé le cloisonnement déjà existant. Tout comme les secteurs du système de santé, les différentes structures qui le composent se sont dotées de SIH qui répondent à leurs besoins, mais dont l'évolution a été faite indépendamment de celle des SIH des autres structures. Certaines structures possèdent même plusieurs SIH, chacun destiné à ne gérer qu'une partie du périmètre d'intervention de la structure. Malheureusement, cette indépendance de développement a fait que la

10. *Les défis de l'accompagnement du grand âge. Perspectives internationales.* (Centre d'Analyse Stratégique, 2011)

plupart de ces systèmes ne communiquent pas entre eux, ce qui impacte la qualité du suivi du patient.

Mais la richesse des données disponibles dans les différents systèmes a depuis plusieurs années suscité l'intérêt des instances de santé, qui ont vu dans leur exploitation un réel potentiel en termes de suivi des patients et d'amélioration de leur prise en charge. Cet intérêt a permis à plusieurs chantiers de voir le jour, notamment autour de la centralisation des données relatives aux patients. Parmi eux, le Programme de Médicalisation des Systèmes d'Information ou *PMSI*, est certainement le plus connu et le plus mature aujourd'hui. Le PMSI est issu de la réforme du système de santé qui vise à réduire les inégalités de ressources entre les établissements de santé. Les articles L. 6113-7 et L. 6113-8 du code de la santé publique stipulent que les établissements de santé publics et privés doivent procéder à l'analyse de leur activité médicale et transmettre aux services de l'État et à l'Assurance maladie les informations relatives à leurs moyens de fonctionnement et à leur activité. Le PMSI contient les données, aussi bien médicales qu'administratives, issues de plusieurs systèmes et relatives aux patients de toute la France. A titre indicatif, dans la période [2006-2015] (soit 10 ans), 12,5 millions de patients ont été hospitalisés et 25,4 millions de séjours ont été enregistrés en moyenne par an. Face à cette quantité de données rassemblées, le PMSI est aujourd'hui ce qu'il y a de plus abouti et ce qui correspond le plus à nos besoins pour une exploitation pertinente. Cependant, plusieurs limitations viennent entraver son utilisation. En nous inspirant de [Prodel, 2017], nous en constatons trois principales.

La première limite du PMSI se trouve dans le **périmètre des données** qu'il contient. D'une part, ces données ne concernent que l'hôpital et non les autres structures. Les différentes prises en charge ne font état que des interventions hospitalières et pourront contenir des "trous" liés aux prises en charge extérieures, pouvant aller jusqu'à plusieurs mois dans le cas de maladies chroniques par exemple. D'autre part, le PMSI reste une base de données différente des systèmes d'informations hospitaliers. Entre autres, il ne contient pas d'informations relatives aux examens médicaux tels que les résultats de tests biologiques ou d'imagerie.

La seconde limitation du PMSI se trouve dans la **qualité des données** qu'il contient. Depuis 2005, le PMSI est utilisé pour l'implémentation de la Tarification à l'Activité, ou *T2A*. Jusque là assuré par une dotation globale forfaitaire versée à chaque établissement, le financement des hôpitaux dépend désormais du nombre et de la nature des actes et séjours réalisés. Ce changement profond dans le mode de financement a eu l'effet un peu pervers de pousser le personnel médical à ne renseigner le PMSI que dans un soucis de tarification des activités. Bien que les actes les plus importants soient nécessaires à cette tarification et donc bien enregistrés, d'autres informations semblent négligées. Dans [Prodel, 2017], l'auteur prend l'exemple d'un patient hospitalisé suite à un Accident Vasculaire Cérébral (AVC), patient pour lequel le diabète n'aurait pas été mentionné car n'impactant pas la consommation de ressources liée au traitement de l'AVC. De plus, la plupart des différents SIH utilisés aujourd'hui ne font pas partie des *Process-Aware Information Systems*. Les Process-Aware Information Systems (PAISs) sont une catégorie de SI qui supportent l'exécution de processus, et ne contiennent pas juste des données isolées. Les ERP (Enterprise Resource Planning), les outils de WFM (Workflow Management) ou encore de CRM (Customer Relationship Management) sont des exemples de PAIS. Dans [van der Aalst, 2016], l'auteur insiste sur le fait que ces systèmes ont en commun la connaissance des processus pour lesquels ils sont utilisés. Les systèmes de type base de données n'ont pas connaissance de ces processus, et ne peuvent donc être exploités efficacement pour les

modéliser ou les contrôler. **Il est donc difficile d'extraire une logique de parcours dans les données disponibles dans le PMSI**.

La troisième limitation relative à l'exploitation du PMSI se trouve simplement dans l'**accès à cette base de données**. Le PMSI représente la partie hospitalière d'une banque de données bien plus importante, le Système National d'Information Inter-Régimes de l'Assurance Maladie (SNIIRAM). Le SNIIRAM est sans doute une des banques de données de santé les plus importantes au monde, pesant près d'un demi pétaoctet. Pour des raisons de risques liés à la mauvaise utilisation de données sensibles, l'accès à cette banque est évidemment très limité, même pour des travaux de recherche. Bien que le périmètre restreint du PMSI autorise plus de souplesse vis-à-vis de l'exploitation de ses données, une accréditation doit être tout de même accordée par la Commission Nationale de l'Informatique et des Libertés (CNIL), procédure administrative qui peut s'avérer être très longue.

Nous avons montré que depuis plusieurs années, les dysfonctionnements du système de santé se renforçaient. Leurs origines sont multiples, provenant aussi bien de limites organisationnelles internes au système, que causés par l'évolution de la société et de ses caractéristiques. Enfin, l'insertion de la santé dans le domaine du numérique a aussi contribué à accroître les dysfonctionnements présents. Les problématiques soulevées dans cette section sont le point de départ de nos travaux. La section suivante a pour objectif de présenter le contexte de recherche ainsi que le positionnement scientifique que nous avons adopté pour répondre à ces problématiques.

1.3 Contexte de la thèse

Les dysfonctionnements organisationnels, l'évolution des maladies et des caractéristiques du territoire rendent le système de santé fragile au niveau opérationnel. Même si elles rencontrent des difficultés, les différentes instances de santé continuent de tout mettre en œuvre pour faire du numérique un socle solide sur lequel s'appuyer pour pallier ces dysfonctionnements et améliorer les prises en charge. Dans cette dynamique, la région Auvergne a été amenée à conduire un certain nombre de "chantiers" en matière d'évaluation et d'amélioration de la performance de l'organisation des prestations de santé, notamment en direction des personnes âgées. Dans l'optique de prendre part à cette dynamique, nos travaux s'inscrivent dans le projet **SaAge**, fruit d'une collaboration régionale entre établissements de santé, institutions publiques et laboratoires universitaires. Après avoir présenté ce projet dans la sous-section 1.3.1, nous détaillons dans la sous-section 1.3.2 les thématiques de recherche que nous avons décidé d'approfondir dans nos travaux afin de répondre aux spécificités du projet. Enfin, dans la sous-section 1.3.3, nous donnerons un aperçu des objectifs visés par nos travaux, d'un point de vue scientifique mais aussi pratique.

1.3.1 Projet SaAge

Financé en partie par la région Auvergne, le **projet SaAge** a vu le jour dans le but **d'analyser les différents parcours de santé de la personne âgée sur la région Auvergne**. Ce projet, soutenu par l'**Agence Régionale de Santé Auvergne**, fait l'objet de deux thèses de doctorat, associant deux laboratoires de recherche, afin de considérer la problématique selon plusieurs perspectives. D'une part, le **Laboratoire d'Informa-**

tique, de Modélisation et d'Optimisation des Systèmes (LIMOS CNRS UMR 6158) s'appuiera sur son expertise pour mettre en œuvre des méthodes de modélisation. D'autre part, le **Centre de Recherche Clermontois en Gestion et Management** (CRCGM, EA 3849) s'appliquera à comprendre comment s'organisent les structures de santé dans la prise en charge des personnes âgées vivant en Auvergne. Alors que la thèse en sciences de gestion s'intéressera aux relations formelles et informelles de la problématique au travers d'enquêtes et d'entretiens, les travaux en sciences de l'ingénieur présentés dans ce manuscrit se concentrent sur l'analyse des parcours.

Les résultats escomptés par le projet SaAge pour la partie sciences de l'ingénieur sont multiples. Il sera nécessaire de faire **un état des lieux** intégré du système, à l'aide de **cartographies** telles que celle présentée dans la section 1.1.1. Un autre objectif consiste à **détecter** des segments peu ou mal pris en charge sur le territoire, ou à l'inverse les prises en charge qui sont particulièrement efficaces.

L'étendue et la souplesse des objectifs visés permettent d'envisager la problématique sous différents angles. En effet, les problématiques de modélisation et plus particulièrement dans la santé ont fait l'objet de nombreux travaux ces dernières années. Dans le but de répondre au mieux aux attentes des différents collaborateurs du projet SaAge, nous présentons dans la sous-section suivante les thématiques de recherche que nous allons aborder dans ce manuscrit.

1.3.2 Thématiques de recherche

Le contrôle des parcours patients joue un rôle important, tant au niveau économique que sociétal, dans les grandes organisations comme peuvent l'être les Centres Hospitaliers. Cependant, nous faisons face à la difficulté de formaliser de bout en bout ces différents parcours du fait d'un système de santé en constante évolution et d'un cloisonnement marqués des professions et des systèmes d'information. Les travaux relatés dans ce manuscrit s'inscrivent dans cette volonté de mettre en lumière les différents parcours de santé. Dans un premier temps, nous allons présenter les caractéristiques de ces processus de santé. L'étude de ces processus est principalement abordée via un exercice de modélisation, dont nous dressons les enjeux dans un second temps.

1.3.2.1 Le parcours de soins comme processus complexe

Comme nous avons pu le voir dans les sections précédentes, le système de santé est une **organisation particulièrement complexe**. La diversité de ses acteurs, de leurs statuts ou de leurs périmètres d'intervention ont été autant de sujet d'études dans le monde de la recherche. Dans ce manuscrit, nous nous intéressons aux parcours de santé. Par conséquent, nous mettons le doigt sur la catégorie à laquelle ils appartiennent, celle des *processus de santé*. Les *processus*, ou *processus métier* selon le contexte, sont un sujet d'étude très vaste et beaucoup de travaux se sont penchés sur leurs concept et enjeux. Par conséquence, il existe un grand nombre de définitions, chacune avec ses spécificités propres, en fonction de l'orientation de recherche qu'ont voulu donner les différents auteurs. Dans le cadre de nos travaux, nous retenons les caractéristiques à l'intersection de toutes ces définitions en nous inspirant de [Cardoso and van der Aalst, 2009] qui définissent un processus métier comme "*un ensemble d'une ou plusieurs activités coordonnées, de sorte qu'ensemble, elles atteignent un objectif métier, habituellement au sein d'une structure organisationnelle*".

Dans le domaine de la santé, les processus sont par essence très différents de ceux que nous pouvons rencontrer dans d'autres domaines tels que celui de l'industrie par exemple [Poulymenopoulou et al., 2003]. En effet, les processus de santé sont reconnus pour être de nature instable, car très soumis aux phénomènes aléatoires. Dans [Rebuge and Ferreira, 2012], les auteurs distinguent deux catégories de processus de santé, les *processus cliniques* d'une part, et les *processus organisationnels* d'autre part. Les processus cliniques sont directement liés aux patients et sont exécutés selon un cycle diagnostic-traitement. Ce cycle dépend fortement des informations collectées relatives à un patient précis et de l'expertise médicale des intervenants prenant les décisions liées au traitement de ce patient. Les processus organisationnels quant à eux, sont des processus "génériques", qui servent de support aux processus cliniques. Ils ne dépendent pas d'un patient ou traitement particulier mais ont pour objectif de coordonner les différents acteurs impliqués.

Quelle que soit la catégorie à laquelle ils appartiennent, l'environnement dans lequel les processus de santé sont exécutés est en perpétuel changement. Le domaine de la santé et les processus qui le composent possèdent des caractéristiques particulières découlant de leur nature dynamique complexe, et pluri-disciplinaire. Dans [Rebuge and Ferreira, 2012], les auteurs recensent quelques unes de ces caractéristiques.

Tout d'abord, les processus de santé sont très *dynamiques*. Les changements qui s'opèrent trouvent leurs origines aussi bien dans l'établissement de nouvelles procédures administratives, que dans les avancées technologiques ou la découverte de nouveaux médicaments. De plus, la médecine est en continuelle évolution, avec la mise au point de nouveaux traitements qui rendent obsolètes ceux en cours. Enfin, de nouvelles maladies apparaissent et celles existantes sont amenées à changer, comme nous l'avons expliqué dans la sous-section 1.2.1. Ces perturbations peuvent nécessiter une réorganisation ou l'implémentation de nouveaux processus.

Au-delà de leur aspect changeant, les processus de santé sont aussi très *complexes*. Cette complexité se retrouve à plusieurs niveaux. La prise de décisions en milieu médical résulte de l'interprétation de données spécifiques aux patients en fonction des connaissances médicales des acteurs. Ces connaissances sont un savant mélange de procédures, de bonnes pratiques et d'expériences personnelles. Ajouté à cela, les informations utilisées pour prendre ces décisions sont de nos jours disponibles en grandes quantités et sont de nature très diverses (rapports, résultats d'examens, ...). Enfin, à traitements et médicaments similaires, chaque patient réagit différemment et des complications peuvent survenir, rendant les processus de santé, de manière générale, imprévisibles.

Ensuite, comme nous l'avons évoqué dans la section 1.2.2, les processus de santé nécessitent de plus en plus de *pluri-disciplinarité*, comme en témoigne la taille grandissante des réseaux de structures au sein desquels les soins médicaux sont réalisés. Cependant, les efforts requis pour la mise en place de cette transversalité sont entravés entre autres par l'augmentation du nombre d'unités spécialisées.

Enfin, les processus de santé sont *ad hoc*. Ils résultent pour une grande partie de l'interaction des différents acteurs qui ont l'expertise et l'autonomie d'adapter les procédures existantes. Le droit que les praticiens s'accordent de ne pas suivre les recommandations, en fonction du cas qu'ils sont en train de traiter, fait des processus de santé des processus variables dont l'exécution n'est pas déterministe.

Toutes ces caractéristiques jouent un rôle dans le manque de visibilité mis en évidence par les institutions de santé (ARS, HAS). Ce qu'il se passe en termes de prises en charge

est une question à laquelle chaque acteur ne peut que répondre partiellement. Bien sûr, il existe de nombreuses procédures, des préconisations que les professionnels de santé s'efforcent de suivre tant que faire se peut. Mais au quotidien, la nature aléatoire des processus de soins, des interventions médicales et des patients prend le dessus sur la nature par essence peu flexible de ces procédures. Les processus devenant de plus en plus complexes, nécessitant une interaction entre organisations et mobilisant des ressources de natures diverses ont rendu la modélisation des processus essentielles pour un suivi de qualité. L'utilité de la modélisation de processus est multiple. Lors de la construction du modèle, le modélisateur est amené à considérer le processus suivant plusieurs points de vue, rassemblant ainsi un large panel de *connaissances*. Une fois construits, les modèles sont utilisés par les acteurs concernés afin *d'orienter* les discussions et les prises de décisions. Ces modèles peuvent aussi servir de *documentation* pour la standardisation de processus ou encore l'analyse de leur performance.

Il existe plusieurs méthodologies destinées à la modélisation de processus. Nous présentons dans la sous-section suivante les défis qu'elles revêtent.

1.3.2.2 Les challenges de la modélisation

Que les organisations ne voient dans les processus qu'un support de discussions, ou bien un outil permettant des analyses beaucoup plus pertinentes [van der Aalst, 2016], la pléthore de langages et de techniques de modélisation existant aujourd'hui illustre l'importance de la modélisation de processus.

La *recherche opérationnelle* fait partie de ces disciplines qui dépendent beaucoup de la modélisation des processus qu'elle cherche à optimiser. La recherche opérationnelle peut être définie comme "*l'ensemble des méthodes et techniques rationnelles d'analyse et de synthèse des phénomènes d'organisation utilisables pour élaborer de meilleures décisions*" [Faure et al., 2009]. Depuis une dizaine d'années, de nombreux travaux en recherche opérationnelle s'attaquent à des problématiques du domaine de la santé. Ils traitent notamment de la modélisation et l'optimisation des flux patients et ont comme objectifs, par exemple, de diminuer le temps d'attente des patients, d'évaluer leur satisfaction, d'optimiser l'utilisation des ressources ou encore de planifier et ordonnancer un ensemble d'activités [Rodier, 2010, Klement, 2014]. En 2011, [Rais and Viana, 2011] citaient plus de deux cent cinquante articles pour les systèmes de soins en général. L'année précédente, [Cardoen et al., 2010] en citaient plus de cent-vingt sur la planification et l'ordonnancement des blocs opératoires. En 2013, une revue de la littérature à propos de l'ordonnancement des ressources humaines dans le système de santé est réalisée par [Van den Bergh et al., 2013] et recense près de trois cents articles. Cette pléthore d'articles démontre la pertinence des méthodes de recherche opérationnelle pour résoudre les problèmes de modélisation et d'optimisation relatifs au système de santé. Bien que la recherche opérationnelle semble être un bon candidat pour envisager les problématiques soulevées par le projet SaAge, elle souffre du fait que "construire un modèle de qualité est plus un art qu'une science" [van der Aalst, 2016]. En effet, encore aujourd'hui, la modélisation des processus est principalement faite "à la main". Étape par étape, la personne en charge de la modélisation s'entretient avec les experts du-dit processus afin d'en capturer l'essence et le formaliser.

Malheureusement, la plupart des modèles construits en suivant cette méthodologie sont relativement déconnectés de la réalité, et n'en proposent qu'une version idéalisée. Dans la Figure 1.4, nous mettons en évidence un des risques majeurs liés à la construction de

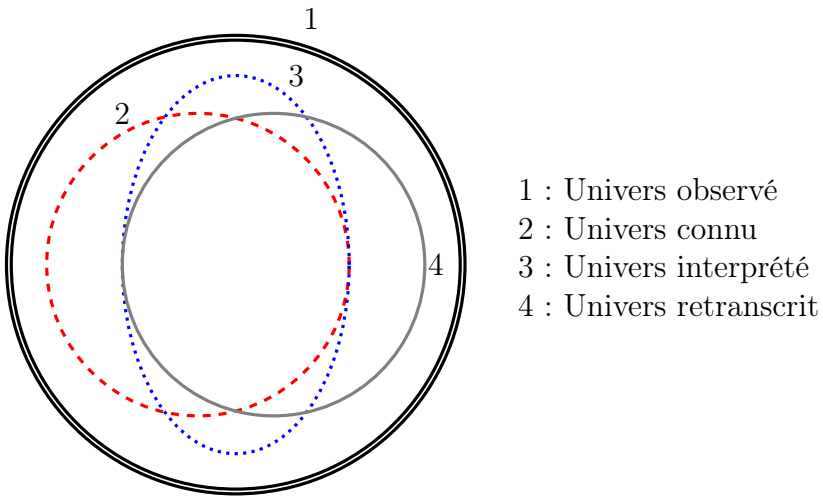


FIGURE 1.4 – Différents niveaux de filtre et d'abstraction de la modélisation à la main.

modèle "à la main" basée sur des entretiens. Le plus grand cercle (1) représente *l'univers observé*, autrement dit toutes les exécutions distinctes du processus étudié qui ont été observées. Dans un contexte de transcription pure et parfaite de la réalité, l'expressivité du modèle construit devrait être identique à cet univers. Cependant plusieurs perturbations viennent entraver cette transcription.

La première et la plus évidente, est représentée par le cercle (2). Cet univers représente *la connaissance limitée* de la personne retranscrivant cette réalité. Cette connaissance limitée peut être expliquée par le fait que plus le processus est complexe, plus il est difficile d'avoir un recul suffisant pour le comprendre. De plus, il est assez rare qu'une personne soit impliquée dans un processus du début à la fin, certains segments de ce processus étant de ce fait plus flous que d'autres. Il y a donc à ce niveau déjà un risque de perte d'information, que le modélisateur cherchera à limiter en augmentant le nombre de personnes interviewées.

Le cercle (3) représente *l'interprétation* du processus par la personne interviewée. Deux personnes intervenant sur le même segment du processus, possédant un niveau de connaissance similaire, peuvent l'interpréter de façon assez différente. "*Ce que je fais n'est pas forcément ce que je crois que je fais*". Nous pouvons expliquer ces différences par le fait que la connaissance des personnes n'est pas limitée au simple processus modélisé. Elles peuvent inconsciemment intégrer des détails issus d'autres processus ou encore extrapoler certaines règles en se basant sur des cas particuliers rarement observés et pas nécessairement similaires à la définition générale qu'elles en donnent. Cela explique pourquoi ce qui est interprété (3) n'est pas seulement qu'un sous-ensemble de ce qui est connu (2).

Enfin, le cercle (4) représente la version véritablement retranscrite par la personne interviewée. Il existe là encore un filtre qui limite la complétude du modèle construit. Il arrive que la personne omette consciemment certains détails, pour cacher des erreurs ou des mauvaises pratiques qui auraient lieu mais qui engageraient leur responsabilité ou encore car la modélisation étant un exercice prêtant à subjectivité, il y a une tendance à rendre les choses plus simples que ce qu'elles ne sont pour des raisons de compréhension. Finalement la version finale du processus ne contient qu'une sous-partie de ce qui est réellement observé par méconnaissance ou filtrage volontaire de l'information, et peut contenir des éléments qui ne l'ont jamais été à cause d'interprétation ou extrapolation inadaptées. Lorsque cette altération de la réalité est importante, elle peut rendre ineffi-

cace/ineffective toute prise de décisions, voire dans le pire des cas dégrader l'exécution des processus qu'ils sont censés représenter.

Une autre limite majeure liée à la construction de modèle "à la main" réside dans l'aspect chronophage de cet exercice. La modélisation de certains processus demande de se positionner à différents niveaux d'abstraction, tantôt très abstraits, tantôt très concrets. Le passage d'un niveau à l'autre ne se fait pas automatiquement et requiert souvent beaucoup de temps. Dans un contexte où les processus sont de plus en plus changeants, il est vital d'avoir des méthodes plus rapides pour la construction des modèles.

Les limites que nous venons de présenter ne sont que des exemples rencontrés par les organisations lorsqu'elles entreprennent une modélisation "à la main". Il est souvent nécessaire d'avoir des analystes et des modélisateurs expérimentés pour permettre la construction de modèles pertinents. Au vu de l'importance accordée à la modélisation des processus dans les organisations, de la capacité aujourd'hui à récolter une grande quantité de données et des limites relatives à une modélisation "à la main", il apparaît comme une nécessité de baser la construction de ces modèles sur des *faits réels* plutôt que sur une interprétation orale de ce qu'il se passe. De cette façon, il sera possible d'avoir un aperçu du processus *réellement* en place dans l'organisation et d'entreprendre des analyses pertinentes. Le *Process Mining* s'inscrit dans cette volonté de modéliser via l'exploitation de données. Discipline récente, le Process Mining a pour but de modéliser, contrôler et améliorer les processus en place dans les organisations en exploitant les "event logs" largement disponibles aujourd'hui dans les systèmes d'information. Un event log contient les informations relatives à l'exécution d'un processus, telles que le nom de l'activité exécutée, la personne responsable ou encore les jour et heure d'exécution. Via l'exploitation de ces données, les techniques de Process Mining visent à analyser les processus existants selon différents points de vue.

Le périmètre d'application et les possibilités offertes par les différentes techniques développées font du Process Mining un candidat très sérieux pour une modélisation pertinente des processus de santé. Cependant, au vu des différentes limitations posées par l'utilisation des données de santé telles que nous les avons présentées dans la sous-section 1.2.3, il apparaît risqué d'envisager le Process Mining pour répondre à la problématique présentée dans ce manuscrit. Et pourtant, c'est bien dans ces risques que nous trouvons la justification de nos travaux.

Nous l'avons vu, aujourd'hui la quantité n'est plus un des critères pénalisant une exploitation pertinente des données de santé. C'est bien leur qualité, au sens large, qui représente son obstacle principal. La plupart des SIH ne font pas partie des PAISs, cette classe de SI qui supportent l'exécution de processus en ayant connaissance de ces processus pour lesquels ils sont utilisés. Cependant, les SIH sont amenés à évoluer. En effet, même si elle a entravé le renseignement du PMSI, la T2A a permis d'enregistrer des données qui ne l'étaient pas jusqu'alors. De plus, les différentes organisations de contrôle s'attachent à rendre les données créées de plus en plus fiables, comme en témoignent par exemple les travaux relatifs au circuit du médicament [Royer, 2014, Angles, 2015].

Par conséquent, les différents SIH constituent une sorte de "boîte noire", dont il est délicat de définir les contours et à travers laquelle il est difficile de distinguer les différents éléments qui la composent. Cependant, si un processus, ou tout du moins un fragment de ce processus, a permis de générer certaines des données présentes dans cette "boîte noire", il serait intéressant de tenter de retrouver ce fragment de processus seulement à partir de ces données, sans connaissance a piori. Est-ce que cela est possible ? Si oui, dans quelle mesure ? Est-ce que l'étude de ces fragments de processus permettrait de détecter

des dysfonctionnements ?

Dans un contexte où le personnel de santé est de plus en plus sollicité, construire des modèles depuis le début devient un exercice à éviter, car très chronophage. Engager des discussions autour de modèles, même approximatifs, mais construits préalablement de façon automatisée, serait un gain de temps non négligeable. Prouver que de tels modèles permettraient d'améliorer la prise en charge des patients mais aussi de faciliter le travail du personnel soignant motiverait ces derniers et justifierait un effort supplémentaire dans le renseignement de bases de santé comme le PMSI.

Enfin, même si initialement motivées par leur application au domaine de la santé, les techniques développées dans ce manuscrit ont pour vocation d'être génériques. En se basant sur différents domaines d'application, nous essaierons au cours de ce manuscrit de montrer la robustesse et la généricité de nos techniques.

Nous avons vu que nos travaux utiliseront les techniques de modélisation, et plus particulièrement de Process Mining, pour analyser les parcours de santé existant. Dans la sous-section suivante, nous présentons en détail les axes de recherche que nous envisageons, et les objectifs visés à travers nos travaux.

1.3.3 Objectifs de la thèse

L'objectif des travaux présentés dans ce manuscrit est de modéliser le parcours de santé de la personne âgée via l'exploitation de données. Bien évidemment, nous nous devons de définir le périmètre de nos travaux. D'une part, cette problématique est bien trop large pour être traitée dans son ensemble. D'autre part, plusieurs travaux ont déjà envisagé les techniques de Process Mining pour la modélisation des processus de santé, comme nous le présentons dans le chapitre 2.

Au-delà de la seule modélisation des parcours de soins, deux pistes de recherche ont attiré notre attention. La première de ces pistes est la *détection de relations de dépendances entre activités*. La seconde piste quant à elle, va consister à considérer un nouveau référentiel, celui de *l'utilité* pour *définir les relations* entre les activités d'un modèle. Nous avons jugé que les challenges scientifiques que ces pistes de recherche proposent et les applications qui peuvent en être faites méritaient que nous nous y attardions plus en détail.

1.3.3.1 Relations entre activités

Pour cette première piste de recherche, nous allons nous intéresser aux relations qui existent entre les activités d'un modèle. Plus particulièrement, nous voulons nous concentrer sur l'impact de l'exécution de certaines activités, notamment en situation de *parallélisme* et de *choix*. Dans une situation de concurrence, il est possible de n'exécuter qu'un sous-ensemble des activités concernées, et ce dans un ordre totalement arbitraire. Dans une situation de parallélisme, même si l'ensemble des activités concernées doit être exécuté, l'ordre d'exécution est également totalement arbitraire. La problématique que nous posons est donc la suivante : dans quelle mesure les choix effectués par le passé, et l'ordre dans lesquels ils ont été effectués, impactent les choix à venir, et l'ordre dans lesquels ils vont être faits ? Quelles sont les *dépendances* qu'il existe entre certaines activités ?

Contribuer à la détection de ces relations de dépendance revêt un intérêt double. D'une part, cela permettrait d'enrichir les techniques existantes et de donner une plus grande précision aux processus modélisés. D'autre part, il serait possible d'extraire de

nouvelles connaissances une fois l'application portée aux parcours de santé. En effet, nous avons présenté dans la section 1.2 le défi de taille qu'est la prise en charge des personnes âgées dans un système de santé et une société en pleine évolution, l'objectif principal étant d'arriver à "absorber" la vague de personnes âgées issues du Baby Boom, le tout dans un contexte de réduction des dépenses de santé. *"Faire plus avec moins"*. Jouant un rôle important dans le bon déroulement des prises en charge, la mobilisation des différentes ressources, humaines comme matérielles, devient de plus en plus un facteur critique et se retrouve au centre des préoccupations : les files d'attente grandissent dans les services d'urgences, la disponibilité des spécialistes allongent les délais de rendez-vous. Le taux d'occupation des lits dans les structures atteint des sommets, des arrangements entre services étant souvent nécessaires pour gérer certains pics. Aujourd'hui, le caractère aléatoire et souvent urgent des parcours de soins implique que la gestion des ressources se fasse principalement "en temps réel", i.e., en fonction de la prochaine activité à exécuter. Or il serait intéressant de voir, en fonction du processus dans lequel nous nous trouvons, si l'exécution d'une ou plusieurs activités conditionne l'exécution d'autres activités plus tard dans le processus. L'idée est de voir dans quelle mesure nous pourrions nous aider de ces relations pour "anticiper" la mobilisation de ressources. Si nous pouvions prévoir un peu plus en amont quelles ressources seront nécessaires dans la suite du processus, cela relativiserait la gestion de ressources dans l'urgence, dans un contexte où il devient important d'optimiser leur efficience.

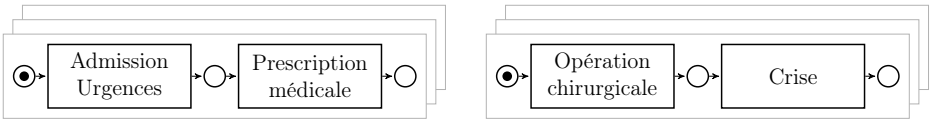
De la même manière, ces dépendances pourraient mettre en évidence que la succession de certaines activités a un impact négatif sur les prises en charge, en termes de durée mais aussi de qualité.

1.3.3.2 Établir la corrélation entre les activités par l'utilité

Aujourd'hui, il existe une variété de mesures utilisées par les techniques de Process Mining pour évaluer la qualité des modèles construits. Parmi elles, la *Fitness* est considérée comme la plus importante. La Fitness est une mesure de qualité qui indique à quel point le modèle est capable de "rejouer" l'event log à partir duquel il a été construit. Nous disons qu'un modèle peut rejouer une séquence d'activités si une des exécutions possibles du modèle est identique à cette séquence. La Fitness indique donc dans quelle mesure le modèle est représentatif de la réalité observée. Par conséquence, afin d'optimiser la valeur de la Fitness, les différentes techniques vont chercher à modéliser les activités ainsi que leurs liens en fonction de leur fréquence d'apparition. Plus deux activités A et B apparaissent l'une après l'autre dans les différentes séquences de l'event log, plus il y de chance qu'elles soient toutes les deux présentes dans le modèle et qu'il y ait une relation directe entre elles.

Même si d'autres mesures de qualité sont aussi utilisées pour l'évaluation des modèles, la Fitness en reste la clef de voûte. Ce point de vue général n'est en rien un problème, tant que nous cherchons à avoir une vue d'ensemble sur ce qu'il se passe, via la modélisation des comportements les plus fréquents par exemple.

Cependant, nous nous permettons de remettre en question la pertinence d'une telle modélisation lorsqu'il s'agit de considérer l'individualité des utilisateurs finaux. Nous présentons dans la Figure 1.5a un exemple de comportement qui pourrait être issu d'un parcours de soins dans un hôpital. L'admission d'un patient suivie d'une prescription médicale est un comportement fréquent, mais sera peu intéressant car il n'apporte aucune information qu'un analyste du domaine médical ne sait déjà. Par contre, le comportement



(a) (b)

FIGURE 1.5 – *(a)* Un comportement fréquent mais peu intéressant *(b)* un comportement peu fréquent mais très intéressant.

présenté dans la Figure 1.5b est certes plus rare, mais indique qu'un choc opératoire (opération + crise) s'est produit, ce qui est une information bien plus pertinente si un analyste s'intéresse à la qualité des prises en charge. L'intérêt que va susciter un modèle chez une personne va donc dépendre des connaissances qu'elle va être capable d'extraire de ce modèle. Or, nous constatons dans les modèles existants une limite majeure. Les modèles dont la construction est guidée par la fréquence d'apparition des différents comportements observés n'apportent pas toujours de réponses utiles aux questions posées par les analystes métier dans le but d'améliorer le processus modélisé. Simplement, ce n'est pas parce qu'un comportement est fréquent qu'il est forcément intéressant. Nous souhaiterions donc que les relations entre activités soient plus "modulaires", en prenant en compte le domaine d'application, l'organisation ou encore le rôle de l'analyste dans cette organisation.

En partant d'un même event log, chaque utilisateur doit pouvoir insister sur certains éléments en fonction de l'importance relative qu'il leur donne. Le modèle construit sera "personnalisé", afin qu'il réponde à une problématique bien précise. Dans cette optique, nous introduisons le concept d'*utilité*. Nous nous référons à ce concept d'utilité pour illustrer de manière générique cette "importance relative" que nous voulons apporter à la modélisation des processus. Dans nos travaux, nous verrons deux instanciations de ce concept, la minimisation du *coût* et la maximisation du *profit*.

A travers la détection des relations de dépendance entre activités et la proposition d'un nouveau référentiel pour la définition des relations entre les activités basée sur leur utilité relative, nous voulons permettre l'extraction de nouvelles connaissances. De cette façon, nous répondons aux objectifs du projet SaAge portant sur l'analyse des parcours de santé. Les travaux présentés dans ce manuscrit s'attachent à présenter les différentes approches envisagées pour traiter ces deux problématiques.

Conclusion

Nous avons présenté dans ce chapitre le contexte environnemental et organisationnel dans lequel le système de santé français s'inscrit. La prise en charge de la population âgée devient de plus en plus difficile ; de moins en moins de moyens sont alloués mais la demande ne cesse d'augmenter. Le système de santé doit donc fonctionner de façon de plus en plus efficiente.

Cependant, les freins déjà présents se renforcent avec l'augmentation de cette population âgée, mais aussi avec l'évolution de la société et des maladies. Parallèlement, nous avons constaté que l'ère du numérique pose de nouveaux challenges et que la richesse des informations accessibles constitue une opportunité à saisir pour l'analyse du système de santé.

Face aux limites que nous avons mises en évidence, nous nous appuyons sur la discipline du Process Mining pour répondre à deux problématiques majeures : détecter les

dépendances entre les activités d'un processus de santé, et personnaliser l'importance de ces dépendances en fonction de la personne qui les analyse.

Première partie

Modèles de Processus et Extraction de Connaissances

Introduction de Partie

Dans le chapitre précédent, nous avons pu constater que le système de santé actuel fait face à plusieurs challenges. La société et les pathologies évoluent, impliquant la mise en place de procédures différentes. L'organisation interne, aujourd'hui très en silo, doit être plus flexible et permettre une plus grande transversalité des prises en charge.

Dans le cadre du projet SaAge, nos premiers travaux ont consisté à dresser un état des lieux des acteurs qui interviennent dans les différentes prises en charge ainsi que leurs interactions.

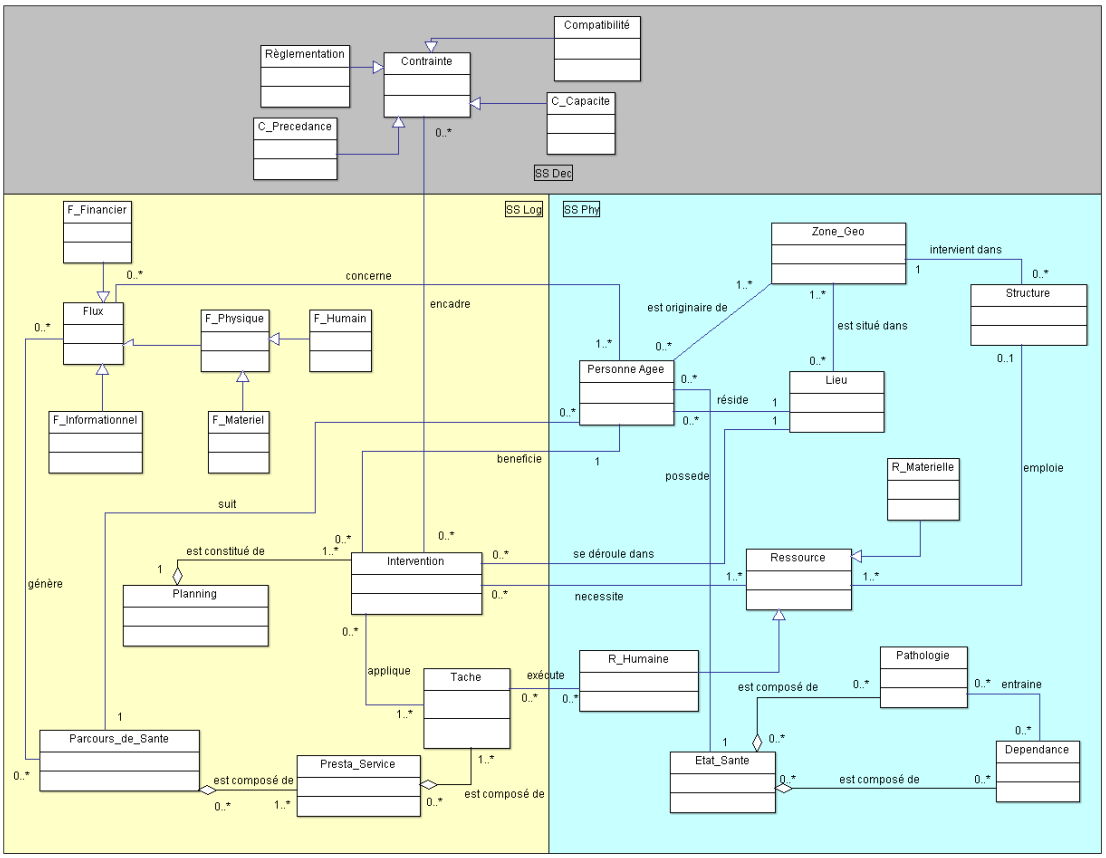


FIGURE 1.6 – Diagramme de classes de la prise en charge d'une personne âgée, réparti en sous-système physique, logique et décisionnel.

En plus de la cartographie introduite dans la Figure 1.1, la Figure 1.6 est un autre

exemple de représentation du système de santé. Dans cet exemple nous utilisons un diagramme de classes dont l'objectif est de modéliser les différents aspects d'une prise en charge. Le diagramme est découpé en trois parties, tel que préconisé par la méthodologie *ASCI* [Gourgand and Kellert, 1991] :

— **sous-système physique** : constitué des entités physiques nécessaires à la réalisation de l'ensemble des services élémentaires. Il comprend l'ensemble des moyens physiques (ressources), leur répartition géographique et leurs interconnexions.

— **sous-système logique** : est constitué des transactions (flux) que le système doit traiter et de l'ensemble des activités concernant le traitement de ces flux.

— **sous-système décisionnel** : est composé de l'ensemble des règles de gestion, d'allocation des ressources et du fonctionnement du système.

Le diagramme présenté illustre bien les points soulevés dans le chapitre précédent dans le sens où nous pouvons constater dans le sous-système physique la multitude de facteurs intervenant dans la réalisation d'une prise en charge. Que ce soit les ressources humaines, matérielles ou bien les structures gérant les prises en charge, beaucoup de règles viennent encadrer les différentes procédures. Ces règles peuvent être définies par des instances de santé, telles que la Haute Autorité de Santé (HAS) ou les Agences Régionales de Santé (ARS), voire les établissements de santé. Cependant, il arrive aussi que ces règles soient informelles, et découlent de l'expérience et des prises de décision individuelles des différents experts du domaine, par exemple dévier de la procédure habituelle en fonction de la réaction d'un patient à son traitement. Comme ces déviances ne sont pas toujours contrôlées, elles peuvent participer à l'existence de dépendances inconnues entre les activités de certains processus. La première partie de ce manuscrit a pour objectif de mettre en évidence ces dépendances.

Cette partie est organisée comme suit. Dans le chapitre 2, nous établissons un état de l'art du process mining. Après avoir présenté en quoi le process mining consiste, nous justifions le choix de cette discipline en donnant un aperçu des applications du process mining faites dans le domaine de la santé. Enfin, parmi les techniques disponibles, certaines possèdent des caractéristiques intéressantes. Nous analysons dans une dernière section dans quelle mesure elles peuvent être utilisées pour traiter nos problématiques.

Dans le chapitre 3, nous présentons les approches proposées pour répondre à nos problématiques. Après avoir introduit les concepts et définitions clefs que nous utilisons tout au long de ce manuscrit, nous proposons *KITE*, une méthodologie générique dont l'objectif est d'utiliser le pouvoir d'expression des modèles de processus pour optimiser l'extraction de connaissances, aussi bien en termes de vitesse d'exécution que de qualité. Dans nos travaux, nous proposons *IDM* et *PRISM*, deux instanciations de la méthodologie pour l'extraction de dépendances intéressantes entre activités dans un processus. Dans cette optique, nous proposons *TWINCLE*, un algorithme que nous avons développé pour l'extraction sous contraintes de règles séquentielles à bas coût.

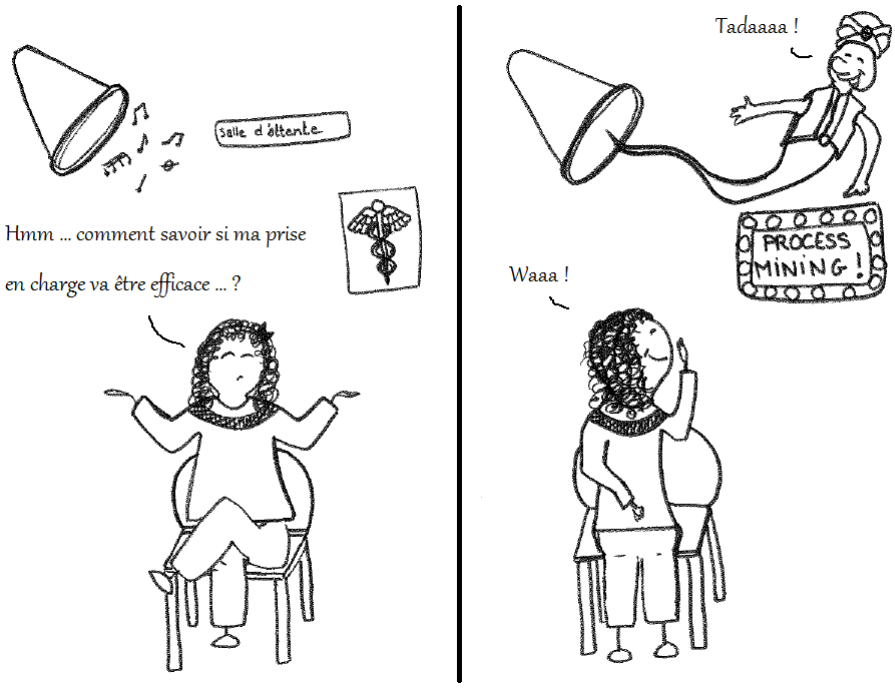
Enfin, nous évaluons les approches proposées dans le chapitre 4. Dans un premier temps, nous effectuons des expérimentations sur l'algorithme TWINCLE, dans l'objectif de mesurer sa capacité à extraire des règles séquentielles à bas coût, ainsi que sa robustesse face à des jeux de données issus de la littérature. Dans un second temps, nous évaluons la capacité de PRISM à extraire des dépendances intéressantes entre les activités d'un processus. Pour cela, nous testons l'instanciation avec deux études de cas : dans la première, nous utilisons un jeu de données généré artificiellement pour représenter un processus

structuré ; dans la seconde nous utilisons un jeu de données extrait du système d'information du Centre Hospitalier du Pays de Craponne-sur-Arzon (CHPCA), représentant un processus peu structuré.

Chapitre 2

Etat de l'Art

Sommaire



Introduction

Ce chapitre a pour objectif de présenter le process mining. Dans la section 2.1, nous introduisons le process mining et présentons un aperçu des analyses qu'il est possible d'effectuer avec les techniques existantes.

Dans la section 2.2, nous détaillons les applications du process mining qui ont été faites dans le domaine de la santé.

Enfin, nous analysons dans la section 2.3 les techniques qui pourraient répondre à tout ou partie des problématiques que nous avons présentées dans le chapitre précédent.

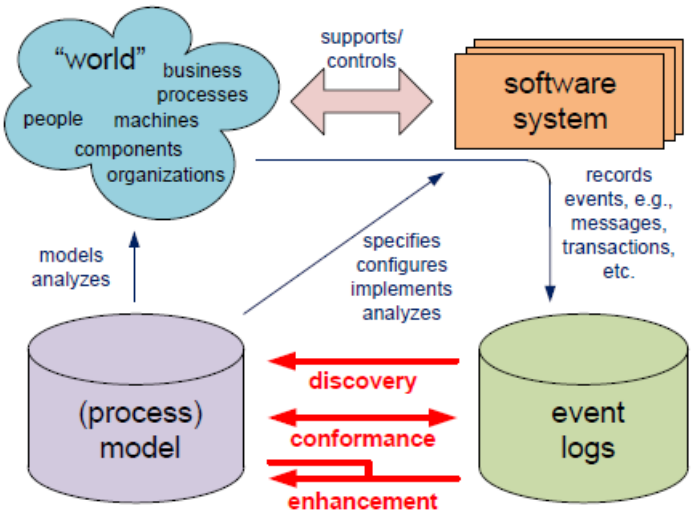


FIGURE 2.1 – Positionnement des trois types de Process Mining : *discovery*, *conformance* et *enhancement* [van der Aalst, 2016].

2.1 Présentation du Process Mining

Les différentes techniques de Process Mining ont pour but de modéliser, contrôler et améliorer les processus opérationnels grâce à l'exploitation d'event logs (ou simplement "log") disponibles dans les systèmes d'information. La figure 2.1 montre le positionnement du Process Mining. Les processus opérationnels sont supportés par différents systèmes qui enregistrent dans des logs tout événement relatif à l'exécution de ces processus. C'est finalement à partir de ces logs que les techniques de process mining construisent des modèles représentant le plus fidèlement possible les processus réels.

L'event log est donc la condition sine qua non de toute technique de process mining. Un log est un ensemble de traces, chacune identifiée par un Id. Une trace est composée d'un ensemble d'events ordonnés dans le temps. Ces events sont composés de différents attributs, certains essentiels, d'autres optionnels, en fonction de la richesse des données disponibles dans les différents systèmes d'information. Ce sont tous ces attributs qui vont permettre d'effectuer des analyses poussées sur les processus en place. Nous présentons dans le Tableau 2.1 un exemple d'event log. L'Id identifie chaque trace. L'attribut "Activité" représente le label de l'activité exécutée et l'attribut "Timestamp" le jour et heure d'exécution. Ces trois attributs sont obligatoires dans un event log, sans lesquels aucune

Id	Activité	Timestamp	Ressource	Coût
1	admission	04-16-2017T14 :11	Clément	50
1	examen	04-16-2017T15 :01	Amélie	150
1	décharge	04-16-2017T16 :24	Thomas	20
2	admission	07-01-2017T10 :00	Jean	50
2	opération	07-01-2017T10 :35	Philippe	260
2	décharge	07-01-2017T10 :59	Monique	60

TABLEAU 2.1 – Exemple d'event log.

analyse n'est possible. Les autres attributs sont des attributs optionnels ; i.e., leur absence n'entraverait pas la construction d'un modèle mais ne permettrait pas d'effectuer des analyses plus poussées. Dans l'exemple, l'attribut "Ressource" présente la ressource qui a effectué l'activité et l'attribut "Coût" présente le coût engendré par l'exécution de cette activité. L'event log présenté dans le Tableau 2.1 est composé de deux traces issues de l'exécution d'un processus de santé, le passage d'un patient dans un établissement. Par exemple, la première trace indique qu'un patient a été admis dans une structure le 16 avril 2017 à 14h11 et a été examiné puis déchargé de cette structure respectivement 50 et 133 minutes plus tard. Chacune des étapes a été effectuée par une ressource différente, et a généré des coûts. Plusieurs ressources peuvent effectuer la même activité et une même ressource peut effectuer différentes activités. La liste d'attributs présentée dans le Tableau 2.1 est loin d'être exhaustive et va dépendre du domaine d'application et du type d'analyse que nous voulons effectuer.

Grâce à ces différents attributs, le spectre couvert par les techniques de Process Mining va permettre d'effectuer des analyses très diverses.

2.1.1 Des types de techniques et des perspectives différents

Même si elles partent d'un même event log, les différentes techniques de Process Mining peuvent être classées selon leur *type* et leur *perspective*, en fonction des objectifs visés par l'analyse effectuée.

La communauté scientifique de Process Mining distingue trois types de techniques :

1. ***Discovery*** : une technique de ce type génère un modèle à partir d'un event log et ce, sans connaissance a priori. Les langages graphiques utilisés comprennent entre autres les réseaux de Petri ou encore les diagrammes BPMN.
2. ***Conformance*** : ce type de techniques compare un modèle (généré par une technique de discovery ou théorique) et un event log afin de voir si la réalité enregistrée dans l'event log est conforme au modèle et vice versa. Ces techniques peuvent être utilisées pour détecter, localiser et expliquer les déviations et en mesurer les impacts.
3. ***Enhancement*** : ces techniques ont pour but d'améliorer un modèle existant en utilisant les informations disponibles dans l'event log. Parmi les techniques d'enhancement, nous distinguons celles qui sont du type *repair* ; i.e., qui modifient le modèle existant pour qu'il reflète mieux la réalité, de celles qui sont du type *extension* ; i.e., qui ajoutent au modèle des informations sur la performance, les ressources ou encore les règles de décision.

Mais si les techniques de Process Mining ont pour objectif principal de modéliser un processus en termes de flux, ce n'est pas leur seule perspective. Il en existe plusieurs, mais quatre sont majoritairement retenues [van der Aalst, 2016], bien qu'il existe des recoupements entre elles :

1. ***Control-flow*** : cette perspective se concentre sur le séquencement des activités afin de modéliser le plus fidèlement possible la réalité observée.
2. ***Organisation*** : cette perspective considère les ressources ayant exécuté les activités afin de les structurer par rôle ou département ou encore pour analyser un réseau social ; i.e., comment les différentes ressources interagissent les unes avec les autres.
3. ***Instance*** : cette perspective analyse les propriétés d'une instance, les caractéristiques qui lui sont propres et qui vont au-delà des activités exécutées, telles que le coût ou le niveau de gravité d'une opération.
4. ***Temps*** : cette perspective se concentre sur le timing et la fréquence des événements. Grâce aux timestamps, il est possible de détecter des points de rupture ou de mesurer le niveau de service.

Pour effectuer ces analyses, plusieurs outils existent. Parmi eux, ProM [1] reste le plus utilisé par la communauté scientifique. Il repose sur un système de plug-ins permettant d'enrichir librement l'outil. ProM en version 6.6 sera l'outil utilisé dans ce manuscrit. Cependant, bien qu'il soit très puissant et modulaire, ProM reste un outil de recherche, et n'est pas très "user-friendly". Pour une utilisation destinée à un public non scientifique, pour les entreprises ou organisations publiques, d'autres outils commerciaux ont été développés, tels que Disco [2], Celonis Process Mining [3], QPR ProcessAnalyzer [4] ou ProcessGold [5] pour ne citer qu'eux. Plus attrayants pour le grand public, ces outils ne contiennent toutefois pas l'ensemble des techniques existantes de Process Mining.

2.1.2 Une pléthore de langages de modélisation

Lorsqu'il s'agit de modéliser un processus, le choix du langage de modélisation apparaît comme une étape cruciale. Nous faisons quelques fois référence à la pléthore de langages existants comme la "nouvelle tour de Babel" [van der Aalst, 2016]. Les langages de modélisation et leurs différences de sémantique n'étant pas au centre de nos travaux, nous ne présenterons ici que (1) les langages les plus utilisés par les techniques de Process Mining et (2) les grandes caractéristiques qui les différencient.

1. **Les réseaux de Petri** [Petri, 1962] :
   Les réseaux de Petri font partie des langages de modélisation les plus étudiés. Leur sémantique, assez simple, est composée de cercles représentant des *places*, de carrés (ou rectangle) représentant des *transitions* et d'arcs orientés liant places et transitions. Une transition peut représenter une activité et lorsqu'elle est déclenchée elle consomme un *jeton* (représenté par un point noir) dans chacune de ses places d'entrée, et produit un jeton dans chacune de ses places de sortie. De cette façon, la

1. http ://www.promtools.org/doku.php
2. https ://fluxicon.com/disco
3. https ://www.celonis.com/
4. https ://www.qpr.com/
5. http ://processgold.com/

distribution des jetons entre les différentes places représente les *états* du modèle, appelés *marquages*. Les marquages initial et final indiquent dans quel état le processus doit commencer et finir.

Un exemple de réseau de Petri est présenté dans la figure 2.2. Dans cet exemple, l'exécution de l'activité A produit deux jetons, un dans p_1, l'autre dans p_2. Ensuite l'exécution de B (respectivement C) consommera le jeton de p_1 (respectivement p_2) et produira un jeton dans p_3 (respectivement p_4). Le marquage final sera atteint avec l'exécution de D, qui consommera un jeton de p_3 et de p_4 ; i.e., l'activité D ne pourra être exécutée que lorsque les activités B et C auront toutes les deux été exécutées.

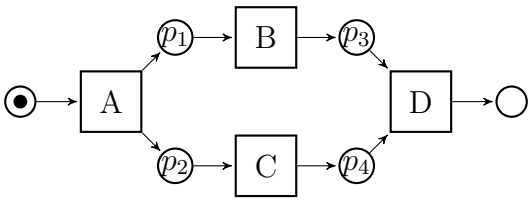


FIGURE 2.2 – Exemple de réseau de Petri.

Bien que leur sémantique soit simple, les réseaux de Petri ont un grand pouvoir d'expression et possèdent des propriétés mathématiques intéressantes. Cependant, ils possèdent quelques limitations lorsqu'il s'agit de modéliser des choix non-exclusifs ; i.e., choisir de déclencher une transition ou une autre mais pas les deux. Cela nécessite un nombre important d'arcs et de transitions silencieuses, ou τ-transitions. Une τ-transition est une transition qui ne représente pas d'activité, mais qui sert simplement à distribuer les jetons dans le réseau de Petri. Mais en augmentant le nombre d'éléments, ces solutions complexifient le modèle et en réduisent la lisibilité.

2. **BPMN** [Object Management Group, 2011] :
Le BPMN (Business Process Model and Notation) est un langage de modélisation très utilisé dans les organisations. Ce langage est riche et possède un grand nombre de symboles. Cependant, les utilisateurs n'utilisent en général qu'un sous-ensemble de ces symboles (environ 10 en moyenne, sur les plus de 50 disponibles [zur Muehlen and Recker, 2008]). Les activités sont représentées par les *tâches* (rectangles avec bords arrondis) et les routages sont effectués via des *passerelles* représentées par des losanges. La Figure 2.3 présente un exemple de diagramme BPMN, identique au réseau de Petri de la Figure 2.2. Après avoir exécuté l'activité A, nous nous retrouvons face à une *passerelle parallèle ouvrante*. Une passerelle ouvrante signifie que le flux entrant va être *divisé*, et une passerelle parallèle indique que tous les chemins sortants de la passerelle doivent être empruntés, mais dans un ordre arbitraire. Dans notre exemple, cela signifie que nous devrons exécuter l'activité B et l'activité C. Nous trouvons une *passerelle parallèle fermante* qui indique qu'il faut attendre tous les flux parallèles devant arriver à la passerelle avant de continuer et enfin exécuter l'activité D pour terminer le processus. Il existe d'autres types de passerelles couramment utilisés, comme la passerelle exclusive (un seul des chemins sortants peut être emprunté) et la passerelle inclusive (un ou plusieurs des chemins sortants peuvent être empruntés, dans un ordre arbitraire). Il est aussi possible de définir des routages en fonction d'événements tels que l'écoulement d'un minuteur ou bien l'arrivée d'un message.

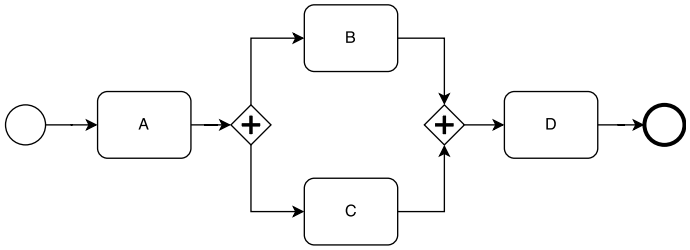


FIGURE 2.3 – Exemple de diagramme BPMN.

3. **Heuristics Nets** [Weijters et al., 2006] :
Les deux précédents langages lient les activités (les transitions pour les réseaux de Petri et les tâches pour BPMN) à d'autres éléments du modèle (les places pour les réseaux de Petri et les passerelles pour BPMN). Cependant, ces éléments ne sont pas présents dans l'event log, ils doivent être inférés en l'analysant. Par conséquent, certains langages tel que les Heuristics nets s'abstraient de ces éléments en reliant les activités directement entre elles. Un exemple d'Heuristics net est présenté dans la Figure 2.4. Les deux arcs sortant de l'activité A sont activés simultanément, et sont donc exécutés en parallèle. La pondération des activités indique le nombre de séquences dans lesquelles l'activité est apparue et la pondération des arcs indique le nombre de séquences dans lesquelles les deux activités liées par un arc sont apparues l'une après l'autre. L'importance des activités et leur relation est aussi caractérisée par la couleur des symboles, plus ils sont foncés plus le nombre d'apparitions est grand. Cette méthode de modélisation implique toutefois que certaines relations sont mal modélisées, comme les choix inclusifs.

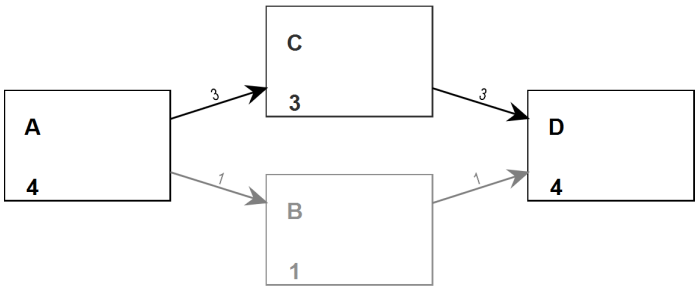


FIGURE 2.4 – Exemple d'Heuristics net.

4. **Fuzzy Models** [Günther and van der Aalst, 2007] :
Comme leur nom l'indique, les Fuzzy models sont "flous" ; i.e., ils ne possèdent pas de sémantique explicite. Les arcs liant les activités indiquent des relations de causalité entre elles et sont caractérisés par leur couleur et leur épaisseur. L'épaisseur d'un arc représente la *corrélation* des activités qu'il lie. Plus un arc est épais, plus les deux activités sont *proches*, i.e., ont des noms similaires ou sont exécutées par la même ressource. La couleur d'un arc indique l'*importance* de la relation. Plus un arc est foncé, plus les deux activités apparaissent l'une après l'autre dans l'event log. La pondération des activités est le résultat d'un calcul avancé et indique l'importance des activités. Un exemple de Fuzzy model est présenté dans la Figure 2.5. La couleur noire des arcs entre les activités B et C d'une part, et D d'autre part, indique qu'à chaque fois que B ou C apparaissent dans une séquence, D apparaît aussi, contrairement aux activités B et C après l'exécution de A. Cependant, les

Fuzzy models n'ont pas de sémantique formelle, ils n'indiquent donc pas clairement comment est structuré le processus.

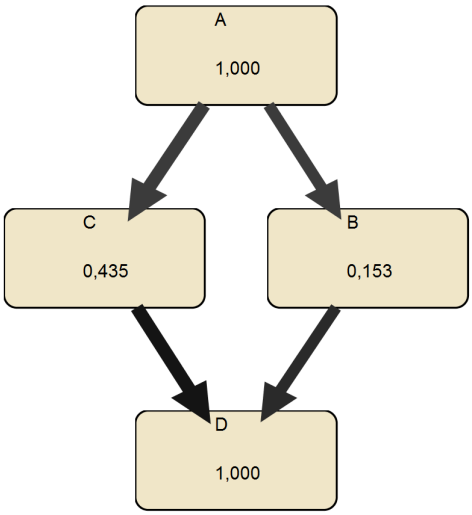


FIGURE 2.5 – Exemple de Fuzzy model.

5. **Process Trees** [Buijs, 2014] :
   Les process trees font partie de la catégorie des *modèles structurés en block*. Un process tree est un modèle hiérarchique dans lequel les noeuds sont des opérateurs et les feuilles sont des activités. Un opérateur sert à structurer les noeuds/feuilles enfants, e.g., l'opérateur XOR ($\times$) indique qu'il n'est possible d'exécuter que l'un de ses noeuds enfants. Un exemple de process tree est présenté dans la Figure 2.6. La racine de l'arbre est un opérateur Séquence ($\rightarrow$), ce qui signifie que ses noeuds enfants doivent être exécutés de gauche à droite. L'activité A, qui est une feuille, est donc exécutée en premier. Ensuite, nous nous trouvons face à un opérateur AND ($\wedge$), qui indique que tous ses noeuds enfants, les activités B et C, doivent être exécutés mais dans un ordre arbitraire. Enfin, le processus termine en exécutant l'activité D.

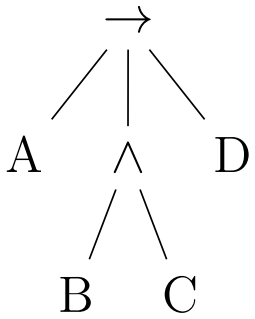


FIGURE 2.6 – Exemple de Process tree.

Bien évidemment, il existe d'autres langages de modélisation tels que les Systèmes de Transitions d'Etats, les Causal Nets [van der Aalst et al., 2011a], YAWL (Yet Another Workflow Language) [Hofstede et al., 2009] ou encore les EPCs (Event-Driven Process Chains) [Scheer, 1994]. Nous ne présentons pas ces langages là pour diverses raisons, certains n'étant utilisés par aucune technique de Process Mining, d'autres ayant un pouvoir

d'expression trop limité ou une syntaxe complexe pour des personnes non expertes. Il est à noter que transformer un modèle d'un langage à un autre est aujourd'hui une tâche assez aisée, dans la mesure où les langages d'origine et de destination possèdent un pouvoir d'expression similaire (e.g. transformer un réseau de Petri en diagramme BPMN).

Comme toute discipline, plusieurs challenges sont à traiter lorsque nous souhaitons mettre en application un projet de process mining [van der Aalst et al., 2011b]. Au niveau des données, les différentes sources, la qualité, mais aussi la quantité des données récoltées sont autant de facteurs impactant la pertinence des analyses faites par les différentes techniques de process mining. Il y a aussi un équilibre à trouver d'une part dans le pouvoir d'expression du langage de modélisation utilisé, d'autre part dans les critères choisis pour évaluer la qualité du modèle. Au niveau des processus, le fait qu'ils puissent changer au cours du temps (*Concept drift* [Bose et al., 2011]), ou bien être exécutés par différentes organisations (*Cross-organizational process mining* [Buijs et al., 2011]) peuvent impacter les analyses effectuées.

De manière générale, le process mining reste un outil pertinent pour la modélisation et l'analyse des processus d'une organisation. Dans la sous-section suivante, nous présentons un aperçu des applications du process mining faites dans le domaine de la santé.

2.2 Le Process Mining dans la santé

Bien qu'à ses débuts, le Process Mining ait été principalement utilisé dans les administrations et les organisations, le domaine de la santé a rapidement attiré l'attention de la communauté scientifique. Petit à petit, de nombreuses techniques ont été appliquées à l'analyse des processus de santé. Ces travaux ont principalement essayé d'analyser ce qu'il s'est passé (e.g. quels sont les parcours types ?), pourquoi ce qu'il s'est passé s'est passé de cette façon (e.g. pourquoi le temps d'attente est aussi long ?) ou encore que va-t'il se passer (e.g. quelle est la probabilité que tel patient dévie du parcours type ?). En nous inspirant et complétant les revues de littérature de [Yang and Su, 2014], [Kurniati et al., 2016], [Rojas et al., 2016] et [Prodel, 2017], nous présentons dans cette sous-section un aperçu des travaux en Process Mining appliqués à la santé.

Les premiers travaux en Process Mining ont été appliqués à des processus très variés, allant de l'imagerie médicale [Lang et al., 2008] aux opérations chirurgicales [Blum et al., 2008], en passant par les opérations de machines à rayons X [Günther et al., 2008] ou encore les soins dentaires [Mans et al., 2012]. Cette diversité démontre le caractère pluridisciplinaire des techniques de Process Mining, bien que les domaines les plus étudiés soient ceux de la chirurgie et de l'oncologie [Rojas et al., 2016].

Cependant, ces travaux se sont majoritairement concentrés sur l'application de techniques de type *discovery*. Dans [Rojas et al., 2016], les auteurs soulignent le fait que parmi les 74 travaux recensés et publiés entre 2007 et 2016, près de la moitié (46%) ont appliqué l'*Heuristic Miner* ou le *Fuzzy Miner*. L'Heuristic Miner [Weijters et al., 2006] est une technique qui génère un Heuristics net tout en limitant les perturbations liées au *bruit* présent dans les event log ; i.e., des données qui n'ont pas lieu d'être et résultant d'une mauvaise manipulation du système par exemple. Le Fuzzy Miner [Günther and van der Aalst, 2007] quant à lui, permet de générer des Fuzzy nets à différents niveaux d'abstraction via l'agrégation de certaines activités en clusters, afin de modéliser des processus connus pour être non-structurés. Pour répondre à cette problématique des processus non structurés, une autre technique très utilisée est le Trace Clustering [Song et al., 2009]. Elle

permet de diviser l'event log en plusieurs event logs plus petits et plus homogènes afin de générer des modèles plus simples et plus structurés. D'autres techniques de discovery ont été utilisées, mais dans une moindre mesure, telles que l'Alpha Miner [Weijters et al., 2004], l'Inductive Miner [Leemans et al., 2013] ou le Genetic Miner [de Medeiros Ana K. et al., 2007].

Avec les premiers résultats convaincants, d'autres types de techniques et d'autres perspectives ont été appliqués. Le Conformance Checker [Rozinat and van der Aalst, 2008] est une technique de type *conformance* qui compare un event log et un modèle afin de déterminer si le modèle est capable de rejouer l'event log et à quel point il n'est capable de rejouer que l'event log. Le Performance Sequence Analyser [van der Aalst et al., 2009b] permet d'effectuer une analyse de l'event log et de ses propriétés en détail. Le Social Miner [Song and van der Aalst, 2008] est une technique exploitant la perspective organisationnelle pour générer des réseaux sociaux afin d'analyser les relations existantes entre les ressources mobilisées dans les processus modélisés. Enfin, dans [Rovani et al., 2015] une méthode de Model Repair est proposée pour améliorer un processus clinique théorique avec les observations constatées.

Même si beaucoup de techniques ont été appliquées au domaine de la santé, la majorité sont de type *discovery* et se basent sur une perspective de flux. Dans [Kurniati et al., 2016], les auteurs soulignent le fait que, sur les 37 papiers recensés entre 2008 et 2016 qui ont une application en oncologie, 36 ont au moins modélisé un processus selon une perspective de flux. Même si 27 ont aussi entrepris une analyse selon une perspective de temps ou d'instance, seulement 5 ont analysé les event logs selon une perspective organisationnelle. Bien que quelques études de cas aient été menées en Asie, en Amérique du Nord ou encore en Australie, environ 73% se concentrent en Europe, dont la plupart aux Pays-Bas, en Allemagne et en Belgique. Cela démontre la nécessité d'élargir le rayonnement du Process Mining dans d'autres pays et régions du monde.

Les différents travaux recensés depuis une décennie nous permettent de constater que la santé est un domaine très prometteur en termes d'application des techniques de Process Mining. Elles permettent d'analyser de façon objective les différents processus en utilisant les données disponibles dans les systèmes d'information hospitaliers.

2.3 Analyse de l'existant

Dans cette section, nous nous concentrons sur les techniques existantes qui répondent en partie aux problématiques évoquées dans ce manuscrit. L'impact des choix et de l'ordre dans lequel ils sont effectués a peu été traité dans la littérature en process mining. Un challenge considéré comme majeur par la communauté en process mining, et qui se rapproche de notre problématique, est la détection de "*non-free choice constructs*". Un non-free choice construct est un phénomène mêlant choix et synchronisation ; i.e., choix et synchronisation ne sont pas séparés. L'issue d'un choix à un instant t dans le modèle n'est pas "libre", il est conditionné par un choix effectué plus tôt dans le modèle. La présence de non-free choice constructs influe sur le comportement autorisé par le modèle, et leur détection est donc synonyme de génération de modèles de meilleure qualité. Nous allons analyser deux techniques de discovery, l'α^{++} *Miner* [Wen et al., 2007] et l'*Heuristic Miner* [Weijters et al., 2006], qui sont à notre connaissance parmi les seules techniques disponibles capables de détecter de tels comportements. Nous insistons sur le caractère "disponible" car le *Genetic Miner* [de Medeiros Ana K. et al., 2007] fait aussi partie des

techniques capables de détecter les non-free choice constructs, mais n'est plus disponible dans les outils de Process Mining à cause de mauvaises performances, notamment un temps de calcul trop important.

Id Instance	Séquence
1	⟨A,D,E⟩
2	⟨B,D,F⟩
3	⟨C,D,G⟩

(a) $\mathcal{L}_1$

Id Instance	Séquence
1	⟨A,D,E⟩
2	⟨A,D,F⟩
3	⟨B,D,F⟩
4	⟨C,D,E⟩
5	⟨C,D,F⟩
6	⟨C,D,G⟩

(b) $\mathcal{L}_2$

Id Instance	Séquence
1	⟨A,C,D,F,H⟩
2	⟨A,C,E,F,G⟩
3	⟨A,C,E,F,H⟩
4	⟨B,C,D,F,G⟩
5	⟨B,C,D,F,H⟩
6	⟨B,C,E,F,G⟩

(c) $\mathcal{L}_3$

FIGURE 2.7 – Events logs avec différents types de dépendances.

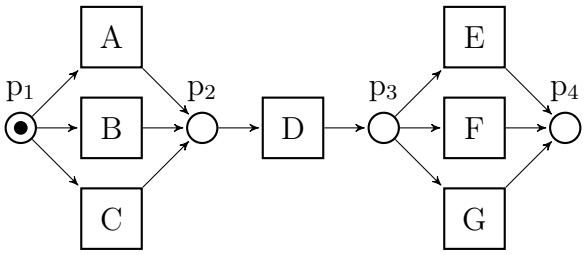


FIGURE 2.8 – Modèle de référence représenté avec un réseau de Petri.

Afin de tester ces deux techniques, nous présentons dans la Figure 2.7 trois différents event logs. Nommons $\mathcal{L}_1$, $\mathcal{L}_2$ et $\mathcal{L}_3$ les logs présentés respectivement dans *(a)*, *(b)* et *(c)*.

$\mathcal{L}_1$ est composé de trois séquences ⟨A,D,E⟩, ⟨B,D,F⟩ et ⟨C,D,G⟩. Cet event log contient des non-free choice constructs, car nous remarquons qu'il existe des dépendances entre les activités A et E, B et F et entre les activités C et G. Le réseau de Petri présenté dans la Figure 2.8 représente le modèle que générerait une technique qui n'est pas capable de détecter les non-free choice constructs, telles que l'α-miner ou le *Fuzzy Miner*. Après avoir exécuté soit A, soit B, soit C, D est exécuté. Ensuite, soit E, soit F, soit G est exécuté et ce, sans contrainte particulière. Nous constatons que ce modèle-là autorise bien plus d'exécutions que ce qui est observé dans l'event log, et n'est donc pas de bonne qualité.

Dans la Figure 2.9, nous présentons deux modèles construits avec *(a)* l'α^{++} miner et *(b)* l'Heuristics miner. Nous constatons que les non-free choice constructs ont bien été

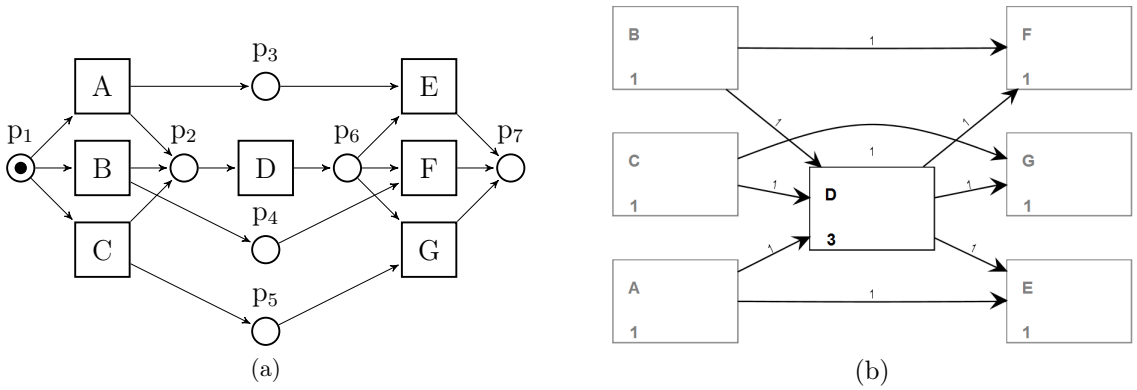


FIGURE 2.9 – Modèles générés avec *(a)* l'α^{++} miner et *(b)* l'Heuristics miner à partir de l'event log $\mathcal{L}_1$.

modélisés. Prenons comme exemple les activités A et E. Dans le réseau de Petri *(a)*, le déclenchement de la transition A va produire deux jetons, un dans p_2 et un dans p_3. Après l'exécution de D, qui consommera un jeton dans p_2 et produira un jeton dans p_6, seule la transition E pourra être déclenchée. En effet, en plus de nécessiter un jeton dans p_6, pour déclencher les transitions F ou G il faut un jeton respectivement dans p_4 ou p_5, ce qui n'est pas le cas ici. Un jeton étant dans p_3 suite au déclenchement de A, il est possible de déclencher E et terminer le processus. Nous constatons la même chose pour les activités B et F et les activités C et G. Le comportement autorisé par le réseau de Petri est donc similaire à celui observé dans l'event log $\mathcal{L}_1$.

Avec un principe similaire et en se basant sur la pondération des arcs, l'Heuristics miner modélise les non-free choice constructs en liant directement les activités concernées. Prenons comme même exemple les activités A et E. Nous savons que chacune n'a été exécutée qu'une fois. Nous savons aussi que l'activité E a été exécutée une fois après A, mais pas forcément directement. Comme E n'a été exécutée qu'une fois après D, et que D a été exécutée une fois après A, nous pouvons en déduire que A et E sont dépendantes l'une de l'autre. En continuant cette logique, il est possible de trouver les autres dépendances.

Cependant, les non-free choice constructs possèdent plusieurs caractéristiques qui font que leur détection ne constitue qu'un sous-problème à celui que nous nous sommes fixé dans nos travaux :

— **bidirectionnelles/unidirectionnelles** : dans un non-free choice construct, la dépendance entre les activités est "bidirectionnelle". Par exemple, pour les activités A et E, nous remarquons que dans l'event log $\mathcal{L}_1$, chaque exécution de A est suivie plus tard d'une exécution de E et inversement, chaque exécution de E est précédée de l'exécution de A. Nous voulons aussi détecter les dépendances "unidirectionnelles". Soit $\mathcal{L}_2$ l'event log présenté dans la Figure 2.7 *(b)*. A la différence de $\mathcal{L}_1$, $\mathcal{L}_2$ ne possède que des dépendances unidirectionnelles. Une première dépendance indique que l'exécution de B implique plus tard l'exécution de F. Une seconde dépendance indique qu'une exécution de G nécessite d'avoir exécuté C plus tôt. Nous remarquons que la réciproque de chacune de ces dépendances n'est pas vraie. La Figure 2.10 présente le réseau de Petri *(a)* et l'Heuristics net *(b)* générés respectivement par l'α^{++} miner et l'Heuristics miner à partir de $\mathcal{L}_2$. Nous constatons que l'α^{++} miner n'est pas capable de détecter les dépendances unidirectionnelles, le réseau de

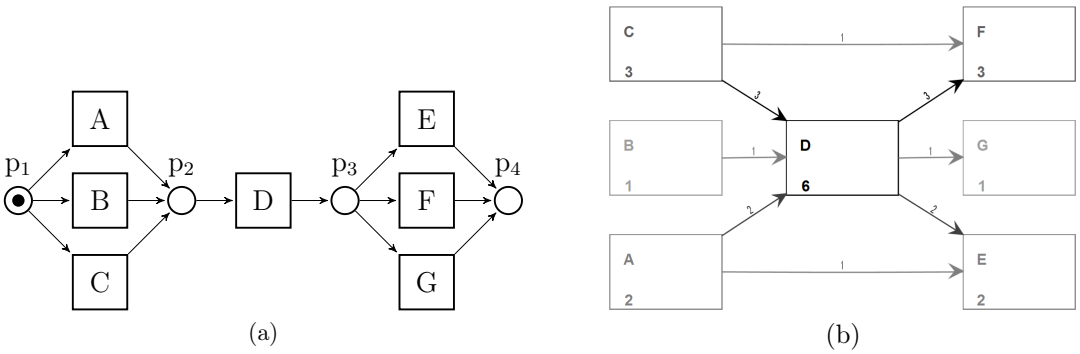


FIGURE 2.10 – Modèles générés avec *(a)* l'α^{++} miner et *(b)* l'Heuristics miner à partir de l'event log $\mathcal{L}_2$.

Petri permettant des comportements non observés. Par contre, l'Heuristics miner est toujours capable de détecter ces dépendances unidirectionnelles. Toutefois, deux limites se posent. La première est que d'un point de vue intuitif, les arcs modélisés entre les activités A et E, et entre les activités C et F semblent indiquer une dépendance unidirectionnelle. Alors que la première est vraie, la seconde ne l'est pas, ce qui porte donc à confusion. La seconde limite est que, bien qu'il soit possible de déduire les dépendances unidirectionnelles à partir du modèle, cela nécessite un exercice de déduction, ce qui n'est pas souhaitable lorsque que nous cherchons la simplicité. De plus, si le modèle est grand, ainsi que le nombre de dépendances, celles-ci ne seront pas explicites pour la personne qui analyse le modèle.

— **totales/fortes** : dans nos travaux, nous voulons introduire une notion de probabilité. La présence de bruit dans les données, l'existence de déviances et la nature aléatoire des processus de santé impliquent qu'il est très rare de voir une dépendance "totale", respectée dans 100% des cas. Nous voudrions pouvoir détecter des dépendances "fortes", dépassant un certain seuil. Là encore, sur le même principe que les dépendances unidirectionnelles, il est possible de constater ces dépendances fortes avec les Heuristics nets, mais cela va nécessiter une fois de plus un exercice de déduction pour y arriver, et complexifier l'analyse.

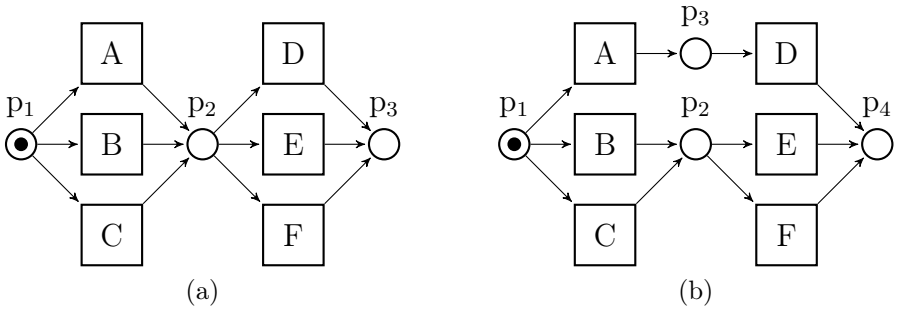


FIGURE 2.11 – *(a)* Modèle avec dépendances directes potentielles et *(b)* modèle avec dépendance totale entre les activités A et D.

— **directes/indirectes** : par définition, pour qu'il y ait un non-free choice construct, une ou plusieurs activités doivent être exécutées entre les activités dépendantes. Or, il arrive dans certaines situations que deux situations de choix soient successives et nécessitent d'être analysées, comme nous le présentons dans la Figure 2.11a. Cependant, dans cette situation la notion de totalité de la dépendance n'a pas lieu d'être. Par exemple, s'il y a une dépendance totale bidirectionnelle entre les activités A et D, signifiant que l'exécution de A est toujours suivie de l'exécution de D et réciproquement, cela ne serait plus considéré comme un choix mais simplement comme une séquence, et le modèle serait celui présenté dans la Figure 2.11b. Pour les dépendances directes, nous nous concentrerons donc uniquement sur les dépendances fortes.

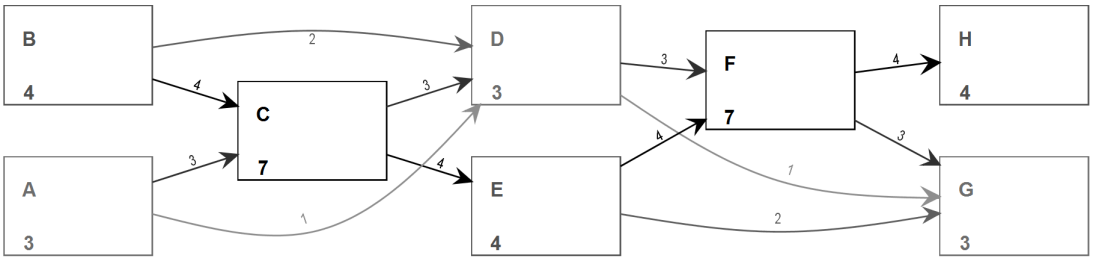


FIGURE 2.12 – Modèles générés avec l'Heuristics miner à partir de l'event log $\mathcal{L}_3$.

— **simples/multiples** : à terme, nous voulons considérer des dépendances concernant plus de deux activités. Considérons $\mathcal{L}_3$ l'event log présenté dans la Figure 2.7 *(c)*. A la différence des event logs $\mathcal{L}_1$ et $\mathcal{L}_2$, $\mathcal{L}_3$ possède une dépendance "multiple" ; i.e., une dépendance qui concerne plus de deux activités. Elle indique que l'exécution de A suivie de l'exécution de D implique l'exécution de H. L'Heuristics net généré avec l'Heuristics miner est présenté dans la Figure 2.12. D'une part, nous constatons que le modèle devient plus complexe à lire, le nombre d'arcs augmentant. D'autre part, comme les dépendances sont détectées seulement entre deux activités dans l'Heuristics miner, les dépendances multiples sont ignorées.

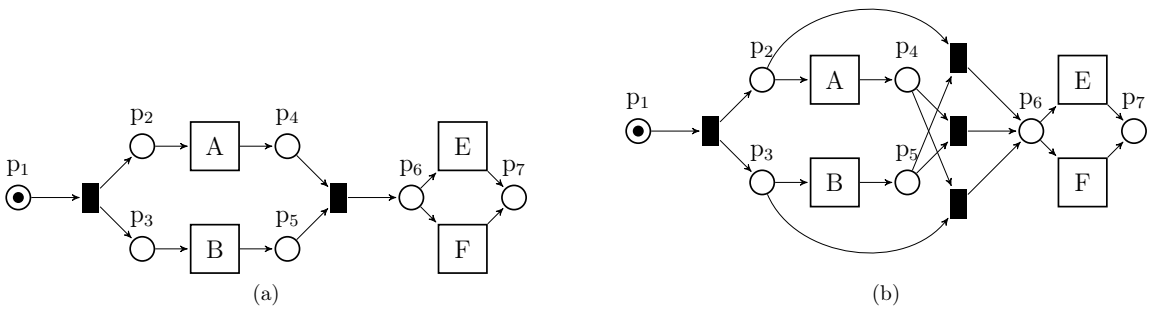


FIGURE 2.13 – *(a)* Modèle avec parallélisme et *(b)* modèle avec choix inclusif.

— **choix/ordre** : comme leur nom l'indique, les non-free choice constructs sont des phénomènes liés à des choix. Dans nos travaux, nous voulons étendre le périmètre des dépendances en considérant aussi l'ordre d'exécution des activités. En effet, il arrive que ce ne soit pas l'exécution (ou non) de certaines activités qui impactent

l'exécution (ou non) d'autres activités, mais l'ordre dans lequel ces activités sont exécutés. Considérons les deux modèles présentés dans la Figure 2.13. Le modèle présenté dans la Figure 2.13a est composé d'un parallélisme entre les activités A et B ; i.e., les deux activités doivent être exécutés mais dans un ordre arbitraire, puis d'un choix exclusif entre les activités E et F. Le modèle présenté dans la Figure 2.13b est composé d'abord d'un choix inclusif entre les activités A et B ; i.e., seulement un sous-ensemble des activités peut être exécuté et ce dans un ordre arbitraire, puis du même choix exclusif que le modèle de la Figure 2.13a. Dans ces situations, nous voulons voir si l'ordre dans lequel certaines activités sont exécutées est vraiment arbitraire et trivial. Par exemple, dans le modèle de la Figure 2.13a, nous voudrions savoir si exécuter A avant B impacte le choix entre l'exécution de E ou de F. Dans le modèle de la Figure 2.13b, nous voudrions savoir si exécuter A et B plutôt que seulement A impacte le choix entre l'exécution de E ou de F.

— **fréquence/utilité** : actuellement, la détection d'une relation entre deux activités a et b se base sur le nombre de fois où l'exécution de b suit celle de a. Dans nos travaux, nous voulons accorder une importance différente aux dépendances existantes en fonction des activités concernées. Cela suppose de ne plus considérer la fréquence comme critère d'importance, mais plutôt un critère d'utilité ; i.e., en utilisant d'autres attributs ; e.g., le coût des events, pour évaluer une dépendance. La raison principale qui nous pousse à remettre en question la mesure de fréquence dans l'évaluation d'une dépendance est le fait que des paramètres autres que le label de l'activité sont susceptibles d'être à l'origine de cette dépendance.

Ce dernier point fait le lien avec la seconde problématique que nous proposons de traiter dans ce manuscrit. Le concept d'utilité tel que nous l'entendons n'a à notre connaissance pas été traité dans les différents travaux en Process Mining. Nous pouvons évoquer les travaux de [de Leoni et al., 2016] qui se rapprochent partiellement de notre problématique. Les auteurs tentent de corréler les caractéristiques d'un modèle. Par exemple, est-ce que dévier du processus "standard" implique des retards dans l'exécution ? Quelle est l'influence de l'expérience d'un praticien sur le parcours de soins d'un patient ? Avec cette approche, il serait possible de considérer comme caractéristiques les différents attributs des événements et prédire par exemple la prochaine activité exécutée. Toutefois, deux limites se posent avec cette méthode. La première réside dans son aspect "prédictif". Pour prédire, il faut définir pour chaque séquence une variable cible ("patient soigné", "en retard") à laquelle nous allons affecter la valeur "vrai" ou "faux" selon si la séquence respecte les conditions de cette variable cible. Autrement dit, (1) il faut savoir ce que nous cherchons et (2) arriver à en définir le périmètre. La deuxième limite est que, même s'il était possible de définir une variable cible, il faudrait en définir pour chaque activité, groupes d'activités ou même groupes d'attributs, ce qui représente un nombre considérable.

Comme notre approche cherche à découvrir ce qu'il se passe, sans connaissance a priori ni questions auxquelles répondre, nous ne pouvons pas nous baser sur ces travaux pour répondre à notre problématique.

Conclusion

Dans ce chapitre, nous avons présenté le process mining et ses objectifs. Nous avons ensuite évoqué les travaux de la littérature en process mining appliqués à la santé. En-

fin, nous avons fait une analyse des techniques existantes qui répondent en partie à la problématique étudiée dans cette première partie.

Nous présentons dans le Tableau 2.2 une grille de lecture récapitulative des techniques que nous avons analysées, par rapport au type du modèle généré par la technique. La grille est construite d'après les critères que nous jugeons importants pour atteindre les différents objectifs fixés par notre problématique.

	Relations de dépendance						
Technique	bi./uni.	tot./fort.	dir./ind.	sim./mult.	ch./ord.	réf.	sim.
α-miner						fr.	
Fuzzy miner						fr.	✓
α^{++}miner			✓			fr.	
Heuristics miner	✓	✓	✓			fr.	

TABLEAU 2.2 – Récapitulatif des limites de l'existant en Process Mining.

Les critères de la grille concernent les relations de dépendance entre activités. Dans ce manuscrit, nous avons évoqué le besoin de détecter et différencier les relations bidirectionnelles (bi.), et unidirectionnelles (uni.), totales (tot.) et fortes (fort.), directes (dir.) et indirectes (ind.) simples (sim.) et multiples (mult.), de choix (ch.) et d'ordre (ord.). Les techniques classiques, à l'image de l'α-miner et du Fuzzy miner, ne sont capables d'en détecter aucune telles que présentées précédemment. Alors que l'α^{++} miner est seulement capable de détecter les relations bidirectionnelles absolues, l'Heuristics miner se rapproche de nos pré-requis en n'étant limité que par la détection des relations multiples. Que ce soit l'α^{++} miner ou l'Heuristics miner, le référentiel (réf.) que ces techniques utilisent dans la détection des relations de dépendance est celui de la fréquence (fr.), ce qui constitue un autre frein majeur pour nos travaux.

Enfin, en fonction du nombre de relations représentées, certains modèles deviennent vite complexes. La simplicité (sim.) d'un modèle est un critère subjectif, mêlant compréhension des symboles d'un modèle ainsi que sa lourdeur. Le Fuzzy miner construit des modèles assez simples, bien qu'ils soient sans sémantique formelle. A l'inverse, les techniques telles que l'α-miner qui modélisent toutes les relations qu'elles détectent complexifie la lecture des modèles et cela peut les rendre illisibles si le processus modélisé est grand.

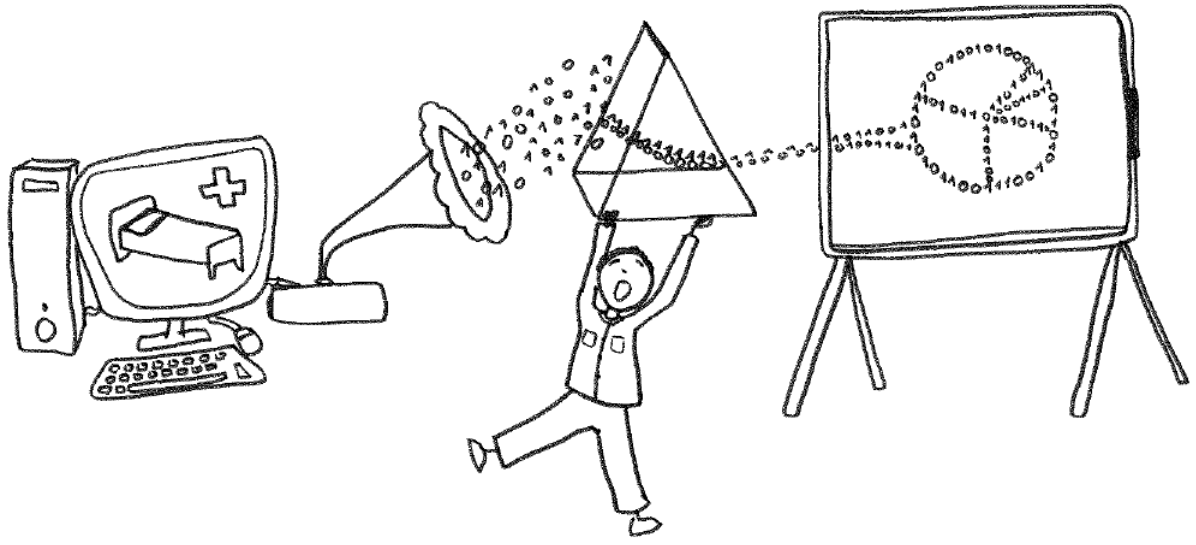
Dans ce chapitre, nous avons pu voir que le process mining est un candidat sérieux lorsqu'il est question d'analyser des processus. De plus, le domaine de la santé étant un domaine dans lequel les processus sont complexes, nous avons pu voir que la littérature s'intéresse de près aux problématiques liées aux différents processus de santé. Cependant, bien que les techniques existantes en process mining possèdent certaines caractéristiques intéressantes, nous remarquons qu'elles restent limitées pour répondre à nos problématiques.

Pour pallier ce manque, nous proposons dans le chapitre suivant une méthodologie pour la détection de tout type de dépendances profitables ; i.e., des dépendances dont l'utilité résulte de l'importance que l'on accorde aux activités concernées.

Chapitre 3

Extraction de connaissances guidée par les modèles de processus

Sommaire



Introduction

Nous avons présenté dans le chapitre 2 comment le process mining a été appliqué dans le domaine de la santé, et avons constaté que les techniques existantes, bien qu'intéressantes, n'étaient pas satisfaisantes pour répondre à nos problématiques. Ce chapitre a pour objectif de présenter une approche développée dans l'optique de pallier ce manque.

Ce chapitre est organisé comme suit. La section 3.1 a pour objectif d'introduire les différents concepts, définitions et notations utilisés en process mining, et que nous allons reprendre tout au long de ce manuscrit.

Dans nos travaux, nous avons défini deux problématiques : la détection de dépendances entre activités et l'évaluation de ces dépendances selon l'importance relative accordée aux activités concernées. Afin de répondre à ces problématiques, nous proposons dans la section 3.2 une méthodologie que nous avons développée pour extraire des relations de dépendance à partir du pouvoir d'expression des modèles de processus. De cette manière, il est possible de (1) définir une importance relative pour les différentes données disponibles et (2) de réduire le temps de calcul nécessaire pour extraire des connaissances intéressantes à partir de ces données. Nous nommons cette méthodologie *KITE*.

Enfin, dans la perspective de répondre aux problématiques posées, nous proposons dans la section 3.3 deux instanciations de la méthodologie KITE pour l'extraction de dépendances profitables. Nous nommons ces instanciations *IDM* et *PRISM*.

3.1 Concepts et définitions

Dans cette partie, nous définissons les concepts de base en process mining, que nous utilisons dans la suite de ce manuscrit.

Définition 1 - Ensemble :
Un ensemble est une collection potentiellement infinie d'éléments. Nous notons un ensemble fini par la liste de ses éléments entre accolades, e.g., un ensemble A contenant les éléments a, b et c est noté {a,b,c}. L'ensemble vide, i.e., l'ensemble qui ne contient aucun élément, est noté $\emptyset$*. Un ensemble non vide est noté* A^+*. Soit* $A = \{a_1, ..., a_n\}$ *un ensemble de taille* $n \in \mathbb{N}$ *, alors* $|A| = n$ *représente la taille de l'ensemble A.*

Dans le reste de ce manuscrit, nous utilisons les lettres majuscules pour représenter les ensembles et les lettres minuscules pour représenter les éléments de ces ensembles.

Définition 2 - Union, Intersection, Différence :
Soit $A = \{a, b, c, d\}$ *et* $B = \{a, c, d, e\}$ *deux ensembles non vides. L'union de A et B, notée* $A \cup B$*, est l'ensemble contenant tous les éléments qui appartiennent à A ou appartiennent à B ; e.g.,* $A \cup B = \{a, b, c, d, e\}$*. L'intersection de A et B, notée* $A \cap B$*, est l'ensemble contenant les éléments qui sont à la fois dans A et dans B ; e.g.,* $A \cap B = \{a, c, d\}$*. La différence entre A et B, notée* $A \backslash B$*, est l'ensemble contenant tous les éléments de A qui n'existent pas dans B ; e.g.,* $A \backslash B = \{b\}$*.*

Définition 3 - Séquence :
Soit A un ensemble, une séquence $s = \langle s(1), ..., s(n) \rangle$ *peut être représentée en listant ses éléments entre chevrons où* $s(i)$ *fait référence au* i^e *élément de la séquence et* $|s| = n$ *représente sa longueur.* $\langle\rangle$ *représente la séquence vide.* A^* *représente toutes les séquences possibles avec les éléments de l'ensemble A, y compris la séquence vide.*

Définition 4 - Concaténation :
La concaténation de deux séquences s_1 et s_2 est notée $s_1 \cdot s_2$. De la même manière, la concaténation d'un élément $a \in A$ et une séquence s est notée $a \cdot s$.

Définition 5 - Fonction totale et partielle :
Soit A et B deux ensembles non vides. Une fonction f de l'ensemble d'entrée A vers l'ensemble de sortie B est une relation où chaque élément de A est associé à un élément de B. Pour toute fonction f, $dom(f)$ représente le domaine (ou ensemble) de définition de f. Pour toute fonction f, si le domaine de définition est équivalent à l'ensemble d'entrée $dom(f)=A$, alors f est dite totale et est notée $f : A \to B$. Si l'ensemble de définition est un sous-ensemble de l'ensemble d'entrée ($dom(f) \subset A$) alors f est dite partielle et est notée $f : A \nrightarrow B$. Une fonction f peut être étendue aux séquences en utilisant la définition récursive suivante : (1) $f(\langle\rangle) = \langle\rangle$; (2) pour toute séquence $s \in A^*$ et tout $x \in A$:

$$f(s \cdot \langle x \rangle) \begin{cases} f(s), & si\ x \notin dom(f), \\ f(s) \cdot \langle f(x) \rangle, & si\ x \in dom(f). \end{cases} \tag{3.1}$$

Définition 6 - Multi-ensemble :
Pour un ensemble A, un multi-ensemble M sur A est une fonction $M : A \nrightarrow \mathbb{N}$. $\mathbb{B}(A)$ représente l'ensemble de tous les multi-ensembles sur A. Par exemple, nous notons $M=[a, b^2]$ un multi-ensemble M sur A où $a, b \in A$, $M(a)=1$, $M(b)=2$ et $M(c)=0$ $\forall c \in A \setminus \{a, b\}$. La taille d'un multi-ensemble M, noté $|M|$, est défini comme $|M| = \sum_{a \in A} M(a)$.

Définition 7 - Projection :
Pour tout ensemble A et $A' \subseteq A$, $s_{\restriction A'}$ représente la projection de la séquence $s \in A^*$ sur A'. Projeter consiste à supprimer de s tous les éléments qui n'apparaissent pas dans A'. Par exemple, $\langle a, a, b, c \rangle_{\restriction \{a,c\}} = \langle a, a, c \rangle$. De la même manière, il est possible de projeter un multi-ensemble M sur un ensemble A', par exemple pour $M=[\langle a, b, c \rangle^3, \langle b, a, c \rangle^2]$, $M \restriction \{a, c\}=[\langle a, c \rangle^5]$.

Définition 8 - Activité :
Une activité est une étape d'un processus et est représentée par un label. Elle est notée $a \in \mathcal{A}$, avec $\mathcal{A}$ l'univers de tous les labels d'activités possibles. Nous notons $\mathcal{A}^\tau$ l'univers des activités auquel nous ajoutons τ, $\mathcal{A}^\tau = \mathcal{A} \cup \{\tau\}$, où $\tau \notin \mathcal{A}$. Le symbole τ représente une activité "silencieuse" ; i.e., une activité qui n'a pas été observée dans les exécutions d'un processus mais qui est représentée dans certains modèles.

Définition 9 - Event :
Soit $\mathcal{E}$ l'univers des events, un event $e \in \mathcal{E}$ est caractérisé par diverses propriétés ; e.g., un event a un timestamp, correspond à une activité, est exécuté par une ressource précise, etc... L'ensemble des propriétés n'est pas fixé, néanmoins nous supposons qu'au moins deux d'entre elles sont le timestamp et l'activité ; i.e., il existe une fonction $\pi_{activity} : \mathcal{E} \to A$ qui assigne à chaque event une activité depuis un ensemble fini d'activités $A \subseteq \mathcal{A}$, et une fonction $\pi_{time} : \mathcal{E} \to \mathcal{T}$ qui affecte chaque event au domaine temporel $\mathcal{T}$. De façon générale, nous écrivons $\pi_p(e)$ pour obtenir la valeur de toute propriété p pour tout event e.

Définition 10 - Trace :
Une trace $\sigma \in \mathcal{E}^*$ est une séquence finie ordonnée et non vide d'events $\sigma = \langle e_1, ..., e_n \rangle$, de sorte que les timestamps entre les events sont non-décroissants ; i.e., pour $1 \leq i < j \leq |\sigma| : \sigma(i) \neq \sigma(j)$ et $\pi_{time}(\sigma(i)) \leq \pi_{time}(\sigma(j))$.

Pour une trace et une propriété données, nous pouvons avoir besoin de créer une séquence composée de la valeur de la propriété pour chaque event de la trace. Pour ce faire, nous adaptons la fonction π_p qui désigne pour un event la valeur de sa propriété p de sorte que nous puissions l'appliquer aux traces. Par exemple, $\pi_{activity}(\sigma)$ est une fonction qui transforme une trace σ en une séquence composée de l'activité de chaque event de la trace. Pour une trace $\sigma = \langle e_1, e_2 \rangle$ avec $\pi_{activity}(e_1) = a$ et $\pi_{activity}(e_2) = b$; $\pi_{activity}(\sigma) = \langle a, b \rangle$.

Les traces peuvent aussi posséder des propriétés. Nous écrivons $\phi_p(\sigma)$ pour obtenir la valeur de toute propriété p pour toute trace σ.

Définition 11 - Event Log :
Soit $\mathcal{C}$ l'univers de toutes les traces possibles, un event log (ou simplement log) $L \in \mathbb{B}(A^*)$ est un multi-ensemble de traces. De façon générale, nous considérons que les events sont uniques ; i.e., le même event ne peut pas apparaître deux fois dans un même log.

3.2 KITE : proposition d'une méthodologie pour une extraction de connaissances guidée

Dans cette section, nous proposons une méthodologie que nous avons développée pour utiliser le pouvoir d'expression des modèles de processus afin d'extraire des connaissances à la fois intéressantes et de façon efficiente. Cette méthodologie se nomme *KITE*.

Après avoir présenté dans la sous-section 3.2.1 les raisons qui nous ont poussés à développer cette méthodologie, nous introduisons ses différentes étapes dans la sous-section 3.2.2.

3.2.1 Motivations

L'objectif de nos travaux dans la première partie de ce manuscrit est de détecter des dépendances potentielles entre les activités d'un processus à partir d'un event log. Si nous élargissons notre problématique, nous pouvons la généraliser à celle d'*extraction de connaissances à partir des données*. De manière générale, l'extraction de connaissances à partir des données (ECD) désigne le processus non trivial d'extraction d'informations implicites précédemment inconnues et potentiellement utiles à partir des données [Frawley et al., 1992]. La *Fouille de Données*, plus connue sous le nom *Data Mining*, fait partie des différentes étapes d'un processus d'ECD, aux côtés de la sélection et du prétraitement des données entre autres. Les connaissances extraites peuvent prendre différentes formes, et peuvent être réparties en deux catégories, celles issues de méthodes *d'apprentissage supervisé* et celles issues de méthodes *d'apprentissage non supervisé*.

— *Apprentissage supervisé* : l'apprentissage supervisé suppose d'avoir des données "étiquetées" ; i.e., il y a une variable *cible* pour chaque séquence de la base de données, les autres variables étant les variables prédictives. Par exemple, si la base de données est composée de résultats de partiels, les notes dans les différentes matières pourraient être les variables prédictives, et la validation ou non du semestre pourrait être la variable cible. Ces techniques cherchent à expliquer la variable cible en fonction des variable prédictives. Parmi les techniques d'apprentissage supervisé, on distingue les techniques de *classification* et les techniques de *régression*. Les techniques de classification, telle que les *arbres de décision*, se basent sur des données

qualitatives alors que les techniques de régression, telle que la *régression linéaire*, se basent sur des données quantitatives.

— *Apprentissage non supervisé* : l'apprentissage non supervisé suppose d'avoir des données "non étiquetées" ; i.e. il n'y a pas de distinction entre variables cibles et prédictives. Ces techniques ne se concentrent pas sur une variable en particulier, mais cherchent à extraire des comportements fréquents dans les données. On peut distinguer les techniques de *clustering* et les techniques d'*extraction de motifs*. L'objectif des techniques de clustering telle que le *k-means clustering*, est de trouver des groupes de séquences qui sont similaires. Les techniques d'extraction de motifs telle que l'*extraction de règles d'association*, visent plutôt à extraire les groupes de données qui apparaissent souvent ensemble dans une base de données.

Notre objectif est de découvrir ce qu'il se passe sur le territoire en termes de processus de soins. De ce fait, de par notre manque d'expertise et pour plus d'objectivité, nous partons sans connaissances a priori du domaine médical. Lorsqu'il s'agit de traiter des données à la recherche de relations intéressantes mais inattendues, les méthodes d'apprentissage non supervisé sont plus qu'indiquées. Parmi les méthodes d'apprentissage non supervisé, les techniques de clustering vont permettre par exemple de définir des sous-ensembles composés de séquences similaires, que l'on pourrait interpréter comme des patients ayant exécuté de la même manière un processus donné. Cela ne permettrait toutefois pas de répondre à nos problématiques. Par contre, les techniques d'extraction de motifs, dont l'objectif est de détecter les groupes de données qui apparaissent ensemble dans une base de données, semblent bien plus adaptées pour répondre à nos problématiques. C'est pour cette raison que nous nous attardons sur ces techniques dans la sous-section suivante.

Cependant, le problème d'extraction de motif soulève plusieurs challenges. Une des préoccupations principales est celle du **temps de calcul**. L'extraction de motifs est connue pour être une tâche très chronophage. Dans un event log, le nombre d'activités et la taille des traces sont parmi les principaux facteurs impactant les temps de calcul. Plus le nombre d'activités distinctes présentes dans le log est grand, plus le nombre de motifs à explorer va croître, et ce très rapidement. Ensuite, la longueur des traces a aussi un impact sur les temps de calculs, la longueur maximum des motifs à explorer étant bornée par la longueur de la plus grande trace du log.

Un deuxième challenge est celui de la **qualité des motifs extraits**. Généralement, un motif est considéré "intéressant" s'il apparait fréquemment. Mais il existe une pléthore de mesures, autres que celle de fréquence, développées dans le but d'affiner l'aspect "intéressant" des motifs [Grissa, 2013]. Pour améliorer la qualité des motifs, il est aussi possible de définir des *contraintes* telles qu'une fenêtre de temps ou un intervalle pour contraindre la longueur des itemsets. Généralement, la définition des contraintes est dépendante du contexte métier dans lequel le processus d'extraction de connaissances est appliqué.

Définition 12 - Profit :
Dans nos travaux, nous utilisons la mesure de "profit" (traduit de l'anglais "utility") pour caractériser l'intérêt que porte un analyste à un certain motif. Un motif est donc "profitable" (traduit de l'anglais "high-utility") s'il apporte un "profit" en termes de connaissances.

Les méthodes utilisées pour l'extraction de motifs intéressants trouvent aussi leur intérêt dans l'**exploitation des résultats**, qui est un autre défi majeur. Une fois l'extraction

effectuée, faut-il encore pouvoir en analyser les résultats. Lorsque des centaines, voire des milliers de motifs sont extraits, il est difficile pour un analyste d'en faire une analyse exhaustive. C'est dans ce sens que l'extraction d'un certain type de motifs, ou l'application de contraintes, aident à réduire la quantité des motifs extraits. D'autres techniques cherchent à réduire cette quantité par l'extraction des k "meilleurs" motifs, dont le concept de "meilleur" est en général défini par la fréquence d'apparition, mais peut l'être aussi par la pondération de différentes mesures d'intérêt.

Pour appliquer les techniques d'extraction de motifs à notre cas d'étude, il faut au préalable prendre en compte ces différents challenges que nous illustrons au travers d'un exemple.

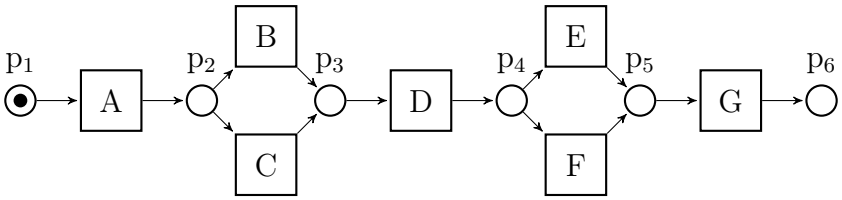


FIGURE 3.1 – Modèle de référence modélisé avec un réseau de Petri.

Considérons le modèle que nous présentons dans la Figure 3.1. Nous l'interprétons comme suit. Après avoir exécuté l'activité A, soit l'activité B, soit l'activité C est exécutée, puis l'activité D. Une nouveau choix est effectué entre exécuter l'activité E ou l'activité F, pour finir avec l'activité G. A partir de ce modèle, nous pouvons faire les deux observations suivantes :

— la première observation est qu'il apparaît clair que certaines associations d'activités ne seront jamais possibles, telles que les activités B et C, ou encore les activités E et F, étant donné qu'elles se trouvent dans des situations de choix exclusifs. Donc si nous ne considérions que les associations possibles, nous pourrions déjà réduire le nombre de candidats à explorer dans l'espace de recherche.

— une seconde observation porte sur l'intérêt de certains motifs. Le problème d'extraction de motifs est défini comme la tâche qui "consiste à extraire des motifs intéressants et inattendus". Si une part de leur caractère intéressant vient du fait qu'ils sont inattendus, alors il se peut que certaines activités ne soient jamais source de motifs intéressants. Dans le modèle de la Figure 3.1, nous pouvons remarquer que l'exécution des activités A, D et G ne sera jamais inattendue, dans la mesure où leur exécution est obligatoire. De ce fait, trouver une dépendance telle que "chaque exécution de A est suivie la plupart du temps d'une exécution de B" n'aurait aucune plus-value, vu que l'on sait que l'exécution de A ne peut avoir d'impact sur les exécutions futures.

En conclusion, nous avons besoin d'une méthode qui permette de réduire l'espace en évitant d'analyser certaines activités. Cela permettrait également de réduire les risques d'extraire des motifs sans aucune valeur ajoutée ou pire, conduisant à de fausses analyses. Ce besoin est évidemment différent d'une simple phase de filtrage, étant donné que le caractère potentiellement "inattendu" des activités ne peut s'anticiper en se basant uniquement sur le log, mais est dépendant de la position des activités dans le modèle.

C'est dans l'objectif de résoudre ce problème que nous proposons la méthodologie *KITE*. Nous détaillons les différentes étapes de cette méthodologie dans la section suivante.

3.2.2 Proposition de la méthodologie

Dans l'optique d'extraire des connaissances plus intéressantes à partir d'un event log, nous proposons dans cette section la méthodologie *KITE* [Dalmas et al., 2017b]. La contribution apportée par cette méthodologie est double : d'une part elle repose sur le pouvoir d'expression des modèles de processus pour réduire l'espace de recherche et ainsi accélérer l'extraction de connaissances, d'autre part car elle permet de se concentrer sur des connaissances plus intéressantes.

Présentée dans la Figure 3.2, *KITE* est une méthodologie générique, dans la mesure où sa conception permet de s'abstraire de tout domaine d'application, l'objectif étant de n'utiliser que les propriétés structurelles des modèles, sans connaissance a priori du processus qu'il représente. L'aspect générique réside aussi dans le fait que les techniques d'extraction de connaissances que l'on peut utiliser ne sont pas limitées à celles d'extraction de motifs ; e.g., il est possible d'envisager des arbres de décisions ou des clusters. Elle est aussi cyclique ; i.e., elle peut être appliquée de façon itérative jusqu'à l'obtention des résultats désirés. Nous détaillons les étapes, au nombre de 4, qui composent cette méthodologie.

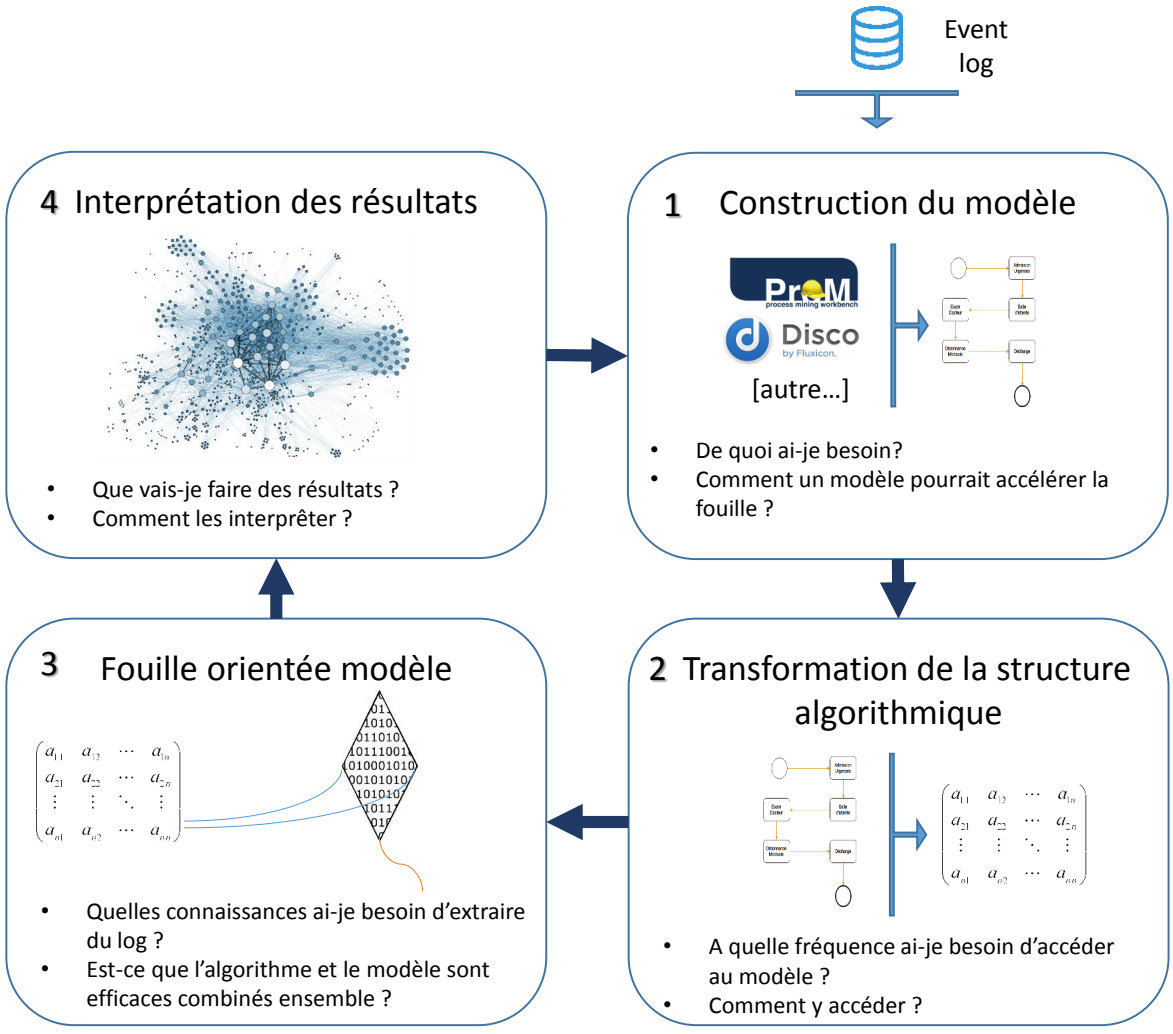


FIGURE 3.2 – Les différentes phases de la méthodologie KITE.

3.2.2.1 Etape #1 : Construction du modèle

Cette première étape consiste à construire un modèle de processus à partir d'un event log, à l'aide d'une technique de process discovery. Le choix de la technique dépend bien sûr du type de modèle que l'on souhaite construire. Ce choix est défini en fonction de différents critères déterminés par le type de connaissances que l'on souhaite extraire mais aussi par les spécificités du domaine d'application. Nous présentons une liste non exhaustive de critères que nous pouvons considérer lors de la construction du modèle :

1. **Faciliter la compréhension du modèle par des non-informaticiens** :
Afin de dialoguer avec différentes parties prenantes, il faut pouvoir utiliser une représentation graphique facilement compréhensible par des non-informaticiens. La technique utilisée doit donc générer un modèle visuellement "user-friendly", ou traduisible dans une notation qui l'est ; accessible à un public varié.

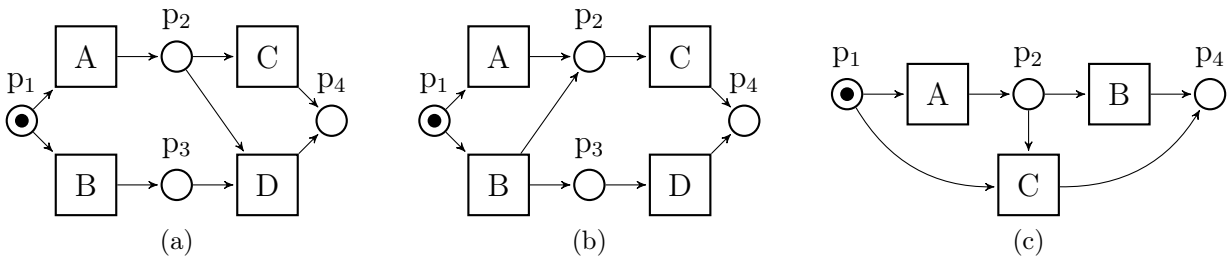


FIGURE 3.3 – Trois réseaux de Petri qui ne respectent pas les conditions de soundness.

2. **Assurer un modèle "sound"** :
Avoir un modèle qui ne possède pas de situations de blocage ; i.e., que l'on peut exécuter sans erreur, est considéré comme un modèle de bonne qualité. La "soundness" d'un modèle évalue cet aspect. Dans le cas des réseaux de Petri, il y a "soundness" si les caractéristiques suivantes sont respectées :

- pour chaque marquage dans lequel le réseau peut se trouver, il doit toujours être possible d'atteindre le marquage final. Le réseau de Petri présenté dans la Figure 3.3a ne respecte pas cette contrainte car si la transition B est déclenchée alors le système est bloqué.
- quand le marquage final est atteint, il ne doit rester aucun jeton dans le réseau. Le réseau de Petri présenté dans la Figure 3.3b ne respecte pas cette contrainte car si les transitions B et D sont déclenchées successivement pour atteindre le marquage final, il restera un jeton dans la place p_2.
- il est possible de déclencher toutes les transitions du réseau. Le réseau de Petri présenté dans la Figure 3.3c ne respecte pas cette contrainte car la transition C ne peut jamais être déclenchée.

Il est donc intéressant que la technique choisie génère un modèle "sound".

3. **Gérer le concept drift** :
Différentes applications ont montré qu'il n'était pas réaliste de supposer que le processus étudié était dans un "état figé" [Bose et al., 2011]. Par exemple, certaines organisations ; e.g., les assurances, peuvent réduire le nombre de dossiers à vérifier en cas de surcharge. En cas de désastre, les hôpitaux ou les banques peuvent changer leurs procédures pour

absorber un pic de prises en charge. Ces changements dans l'exécution d'un processus sont appelés *concept drift*. De tels changements sont présents implicitement dans le log et les détecter peut s'avérer être important si l'on souhaite optimiser le processus modélisé.

4. **Modéliser les différents comportements** :
La nature de certains processus impose que le modèle généré soit représenté dans un langage dont le pouvoir d'expression permet de capturer leur complexité. Les comportements principaux ; e.g., choix, parralélisme, boucle ; doivent donc être modélisables.

5. **Gérer le bruit** :
La notion de *bruit* peut être définie au sens strict ou au sens large. Au sens strict, le bruit désigne les observations présentes dans le log qui ne sont pas représentatives du système ; e.g., les erreurs d'enregistrement. Au sens large, le bruit comprend aussi les comportements peu fréquents. Clairement, il n'est pas possible pour une technique de process discovery de distinguer un comportement rare d'un enregistrement incorrect. Par conséquent, la notion de bruit au sens large sera celle retenue. Être capable de filtrer le bruit est essentiel car cela permet de minimiser la complexité du modèle, tout en modélisant les comportements les plus observés. Dans ce contexte, nous parlons souvent de "modèle 80/20" ; i.e., un modèle capable de décrire 80% des observations qui ne sont en général responsables que de 20% des variations.

6. **Prendre en compte les ressources** :
Lors de la modélisation de certains processus, il arrive que la prise en compte des ressources soit nécessaire pour que l'analyse du processus soit pertinente ; e.g., dans la gestion des blocs obstétricaux [Rodier, 2010]. Par conséquent, il faut considérer un modèle dont la notation permet de modéliser les ressources tel que les réseaux de Petri ou certaines extensions de BPMN.

7. **Considérer les différentes mesures de qualité** :
Lorsqu'une technique génère un modèle, elle se base sur certaines mesures pour évaluer sa qualité. Parmi les mesures disponibles, les plus utilisées pour juger de la qualité d'un modèle sont la *fitness*, la *précision*, la *généralisation* et la *simplicité*. Ces 4 critères de qualité sont définis comme suit [Buijs, 2014, van der Aalst, 2016] :

— **Fitness** :
Un modèle a une bonne fitness s'il autorise le comportement observé dans le log. De manière générale, nous considérons qu'un modèle a une fitness "parfaite" s'il permet de rejouer toutes les traces du log.

— **Précision** :
Un modèle est précis s'il n'autorise pas "trop" de comportements, en comparaison de ceux observés dans le log. Lorsque la précision du modèle n'est pas bonne, il y a *sous-apprentissage* (ou *underfitting*). Le problème de sous-apprentissage est le fait que le modèle est trop permissif ; i.e., il autorise des comportements très différents de ceux observés dans le log.

— **Généralisation** :
La généralisation d'un modèle évalue à quel point les comportements de ce dernier ne se restreignent pas seulement aux comportements observés dans le log. Lorsque la généralisation n'est pas bonne, il y a *sur-apprentissage* (ou *overfitting*). Le problème de sur-apprentissage est le fait que le modèle est trop spécifique à un log donné ; un autre log provenant du même système produirait un modèle totalement différent.

— **Simplicité** :
Le critère de simplicité fait référence au *Rasoir d'Occam* (ou *principe de parcimonie*) : *Pluralitas non est ponenda sine necessitate*, "les entités ne doivent pas être multipliées par delà ce qui est nécessaire". Dans le contexte de la modélisation de processus, cela signifie que le meilleur modèle tend à être le modèle le plus simple pour expliquer les comportements observés dans le log. Ce critère n'est pas nécessairement lié aux comportements observés dans le log ou autorisés par le modèle.

Un bon algorithme doit donc chercher l'équilibre entre précision et généralisation, tout en rejouant le maximum de comportements observés et ce, le plus simplement possible. Atteindre parfaitement cet objectif peut être considéré comme le Saint-Graal des techniques de process discovery.

Lorsque les critères pertinents ont été sélectionnés, nous pouvons passer à l'étape suivante de la méthodologie.

3.2.2.2 Etape #2 : Transformation de la structure algorithmique

Lors de la construction d'un modèle, une technique manipule rarement la notation dans laquelle il est représenté graphiquement. Par exemple, une technique qui produirait un réseau de Petri ne manipule pas des places et des transitions lors de la phase de construction. Chaque technique utilise une structure intermédiaire que nous appelons *structure algorithmique* ; i.e., deux techniques produisant un réseau de Petri peuvent chacune avoir leur propre structure algorithmique. Ce sont ces structures algorithmiques que les techniques vont "traduire" pour générer la notation graphique du modèle.

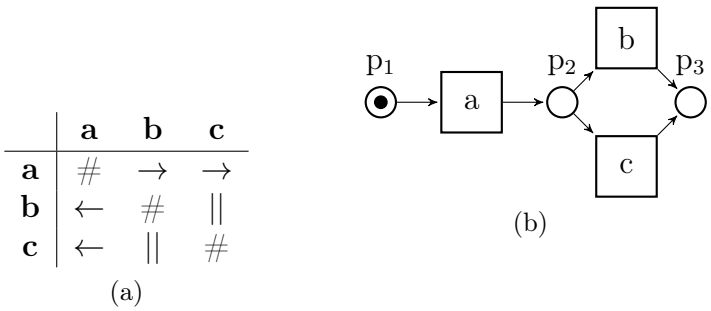


FIGURE 3.4 – (a) Matrice d'empreintes d'un log contenant les activités a, b et c et (b) le réseau de Petri correspondant.

Un exemple de structure construite par un algorithme est la *matrice d'empreintes* construite par l'α-miner. Nous présentons dans la Figure 3.4a un exemple de matrice d'empreinte construite à partir d'un log contenant trois activités a, b et c. Pour deux activités quelconques a et b, $a \rightarrow b$ signifie que l'activité a est directement suivie de l'activité b au moins dans une trace du log et que la réciproque n'est pas vraie ; $a \# b$ signifie que l'activité a n'a jamais été directement suivie de l'activité b et $a || b$ signifie qu'il y a au moins une trace dans laquelle a apparaît juste avant b et au moins une trace dans laquelle b apparaît juste avant a. A partir de cette matrice d'empreintes, l'α-miner traduit chaque comportement pour construire un réseau de Petri, que nous présentons dans la Figure 3.4b.

Dans la méthodologie KITE, c'est sur la structure algorithmique que l'extraction de connaissances va se baser ; mais il se peut que celle utilisée par la technique ne soit pas

adaptée. Par exemple, une matrice d'empreintes permet de connaître les activités qui se suivent, mais pas dans quelle proportion. Un autre exemple est que nous pourrions ne pas être intéressés par certaines activités. L'objectif de cette étape est de réduire, transformer de telles structures algorithmiques afin de permettre à un algorithme de fouille de l'exploiter. En fonction de l'information recherchée et de la fréquence d'accès aux données stockées, une structure sera donc beaucoup plus adaptée qu'une autre pour n'explorer que l'espace de recherche nécessaire par exemple.

3.2.2.3 Etape #3 : Extraction de connaissances guidée

La troisième étape est le cœur de la méthodologie. C'est celle qui consiste à définir la technique de fouille à utiliser. Ce choix se base bien évidemment sur le type de connaissances que nous voulons extraire et comment nous voulons les représenter. La diversité des techniques disponibles permet d'extraire diverses représentations des connaissances : associations, séries temporelles, clusters ou encore à l'aide d'arbres.

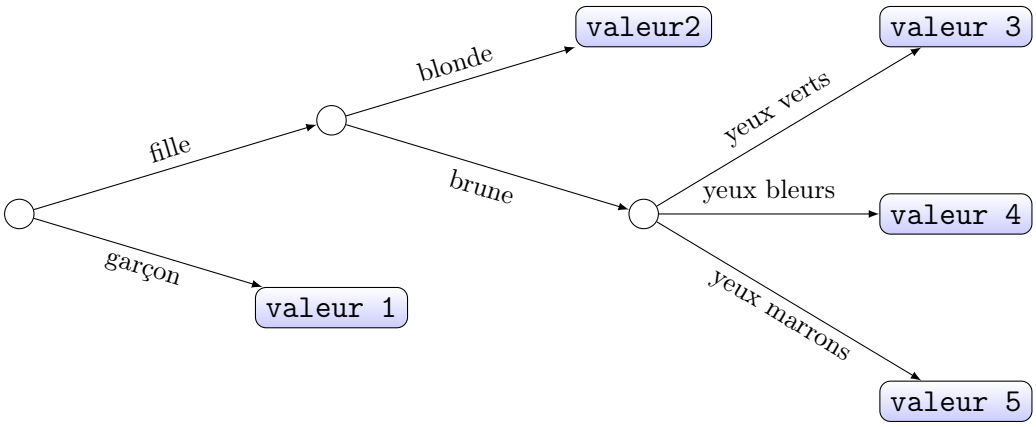


FIGURE 3.5 – Exemple d'arbre de décision.

Nous présentons dans les Figures 3.5 et 3.6 deux exemples d'extractions. La Figure 3.5 montre un *arbre de décision*. Ce type de techniques représente les connaissances extraites sous forme arborescente. Ce sont des techniques *prédictives* ; i.e., à partir d'une base d'apprentissage contenant plusieurs observations, le modèle tente de prédire l'une des variables d'une nouvelle observation. Dans l'exemple, l'arbre de décision prédit une valeur donnée en fonction des caractéristiques physiques d'une personne.

La Figure 3.6 est issu d'une technique de *clustering*. Ce type de techniques représente les connaissances extraites sous forme de *cluster* ; i.e., par groupes homogènes. Ce sont des techniques *descriptives* ; i.e., elles tente de décrire des relations entre les données observées. Dans l'exemple, les différentes traces d'un log peuvent être classées selon deux mesure (Mesure 1 et Mesure 2). On voit que deux clusters se dégagent, et c'est à partir de ce constat-là que l'analyste peut tirer des conclusions pertinentes.

Cette étape permet donc à l'utilisateur de se tourner vers des algorithmes spécifiques afin d'atteindre les objectifs en termes de modèles et de connaissances à extraire. De plus, diverses contraintes peuvent être rajoutées pour respecter des besoins "métier", telles que des fenêtres de temps, de longueur ou encore de complexité. En plus de l'efficacité de la technique appliquée en terme de pertinence de la connaissance extraite, une attention particulière sera portée sur la vérification de la compatibilité entre l'algorithme de fouille et la structure algorithmique employée, en terme de vitesse d'exécution. Par exemple, il

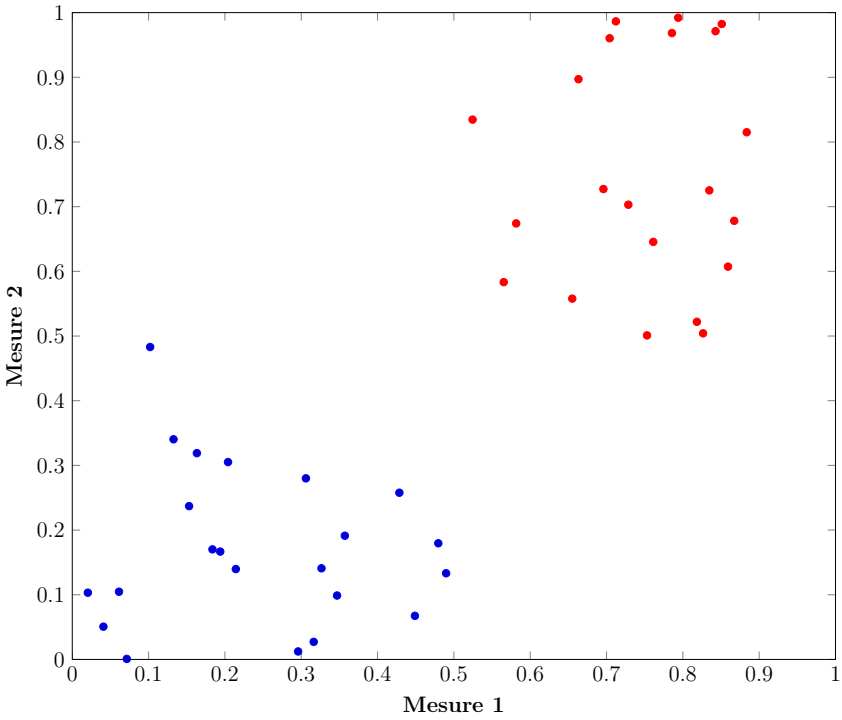


FIGURE 3.6 – Exemple de clustering des différentes traces d'un log en fonction de deux mesures.

se peut qu'utiliser une matrice d'empreintes pour construire un arbre de décision ne soit pas optimal, voire dans le pire des cas pas faisable.

3.2.2.4 Etape #4 : Interprétation des résultats

Cette dernière étape consiste à analyser les résultats de la fouille. Bien qu'elle se situe en fin de cycle, cette étape doit se penser en amont, car c'est elle qui va déterminer les propriétés que le modèle et la structure algorithmique doivent posséder. Le type de connaissances à extraire va de la même façon être défini par les questions auxquelles l'analyste souhaite répondre.

Si l'analyste souhaite connaître les activités souvent exécutées ensemble, il va plutôt se tourner vers des techniques d'extraction de motifs ou de clustering ; s'il souhaite d'avantage comprendre quelles activités impactent la durée d'exécution du processus, les arbres de décisions ou les techniques de régression sont peut-être plus appropriées. L'analyste doit aussi s'interroger sur la manière d'interpréter ces résultats et d'évaluer leur cohérence.

Dans le cas où les objectifs fixés ne seraient pas atteints à ce stade de la méthodologie, son caractère itératif permet un retour à l'étape #1 afin de redéfinir le modèle ou la structure sur lesquels est basé l'algorithme de fouille, voire décider d'une autre technique d'extraction de connaissances plus adaptée.

Dans cette section, nous avons introduit la méthodologie KITE. Dans nos travaux, nous avons proposé deux instanciations de cette méthodologie. La première, que nous appelons *IDM* [Chabrol et al., 2016], traite de l'impact des choix sur l'exécution du processus. Nous avons étendu le périmètre d'application de l'IDM en développant *PRISM*, pour traiter de l'impact des choix et de l'ordre dans lequel ils sont faits sur l'exécution du processus. Nous présentons cette extension dans la section suivante.

3.3 PRISM : proposition d'une instanciation pour la détection de dépendances entre séquences d'activités

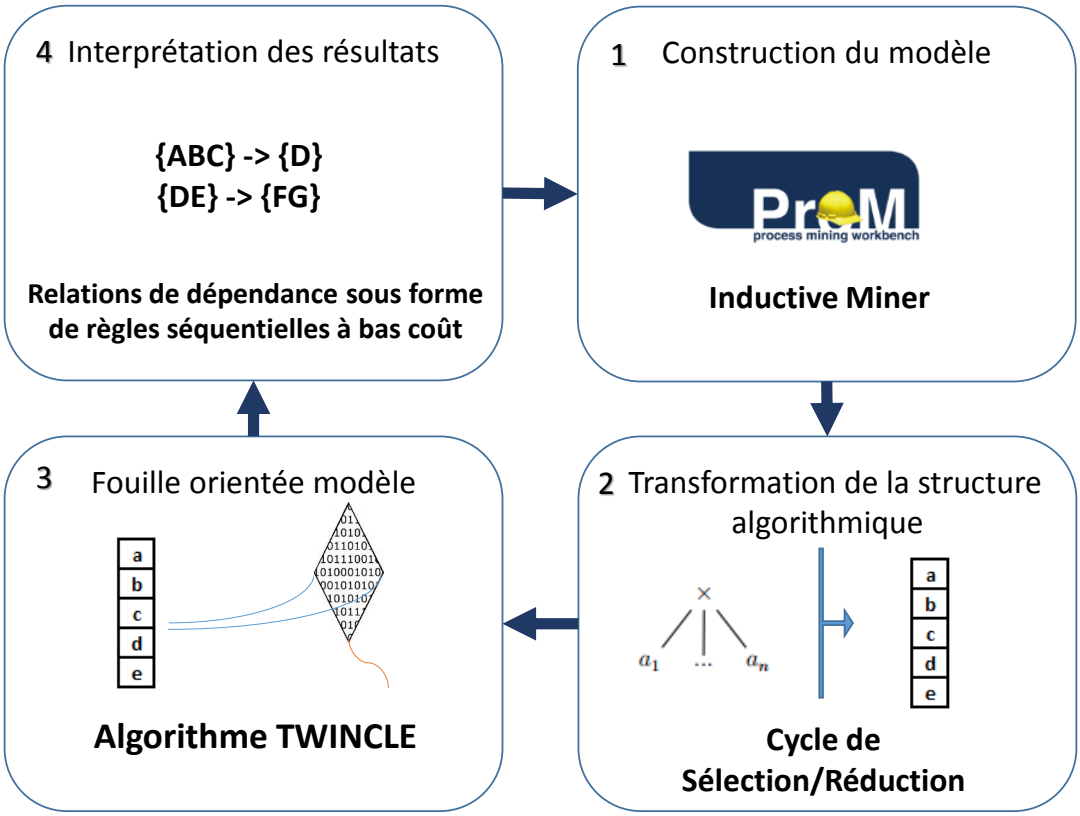


FIGURE 3.7 – Les différentes phases de l'instanciation PRISM.

Dans cette partie, nous introduisons *PRISM* (pour *Process-Integrated Sequential Mining*), une instanciation de la méthodologie KITE pour la détection de relations de dépendance entre activités. Nous présentons le schéma de cette instanciation dans la Figure 3.7. L'étape de construction du modèle sera traitée dans la sous-section 3.3.1. La sous-section 3.3.2 sera par la suite réservée à la transformation de la structure algorithmique dans le but de répondre à notre problématique. Enfin, dans la sous-section 3.3.3, nous présentons *TWINCLE*, un algorithme pour l'extraction de règles séquentielles à bas coût avec contraintes. L'étape relative à l'interprétation des résultats sera traitée dans le chapitre suivant.

3.3.1 Construction du modèle

Cette sous-section présente l'instanciation de l'étape #1 de la méthodologie. Après avoir défini les contraintes que nous jugeons importantes pour notre domaine, nous présentons la technique de process discovery qui, à notre connaissance, respecte le mieux ces contraintes.

3.3.1.1 Choix des contraintes

Pour la première étape, il faut choisir quelle technique utiliser pour construire le modèle. Comme préconisé dans la section précédente, une attention particulière doit être portée sur le type de modèle généré. Comme ce choix dépend des connaissances que l'on cherche à extraire mais aussi des spécificités du domaine d'application ; i.e., la santé dans le cadre de nos travaux, nous avons défini une liste de 5 contraintes que nous nous imposons pour le choix de la technique, et que nous avons définies dans la section précédente :

1. **Faciliter la compréhension du modèle par des non-informaticiens**
2. **Assurer un modèle "sound"**
3. **Modéliser les différents comportements**
4. **Gérer le bruit**
5. **Faire un compromis entre les mesures de Fitness, de Précision, de Généralisation et de Simplicité**

La "soundness" est un prérequis important si nous souhaitons construire un modèle de qualité. Les processus de santé sont des processus complexes, dans lesquels l'aléa a une part importante, il est donc nécessaire de pouvoir modéliser tous les comportements si nous voulons avoir un aperçu représentatif de la réalité. De plus, comme la qualité des données n'est pas aujourd'hui à son maximum, il est nécessaire de gérer le bruit, pour filtrer les potentielles erreurs de saisie par exemple. Aujourd'hui, la pondération entre les 4 mesures de qualité principales est un objectif que la plupart des techniques se fixent, et à juste raison pour assurer un modèle sur lequel il est possible d'extraire des informations pertinentes. Enfin, il est important de pouvoir dialoguer avec des experts du domaine de la santé, pour cela nous avons besoin d'utiliser une notation simple et compréhensible par des non-informaticiens pour représenter graphiquement les modèles.

Dans le but de respecter ces 5 contraintes, notre choix s'est porté sur l'*Inductive Miner* introduit par [Leemans et al., 2013]. L'Inductive Miner est une technique qui, à partir d'un log en entrée, produit un *process tree*. Un process tree est une notation utilisée pour représenter un processus, et est structurée en blocs. Les modèles structurés en blocs obligent que chaque flux ouvrant ait son flux fermant équivalent, par exemple une passerelle ouvrante et fermante du même type pour un diagramme BPMN. De plus, ils n'autorisent pas qu'une dépendance entre ou ne sorte entre une passerelle ouvrante et sa passerelle fermante équivalente. Par conséquent, ces modèles sont assurés d'être sound. Graphiquement, un process tree est simple à comprendre mais n'est pas très "user-friendly", et donc peu adapté pour interagir avec des utilisateurs finaux. Cependant, transformer un process tree dans une autre notation est simple. Parmi les langages de modélisation graphique qui respectent la contrainte d'être facilement compréhensible, BPMN est le candidat que nous avons retenu. BPMN est devenu une des notations graphiques les plus utilisées pour la modélisation de processus métier, et s'est naturellement imposée auprès des professionnels issus de différents domaines [van der Aalst, 2016]. Cette notation est donc accessible à un public varié. L'Inductive Miner construit un process tree de sorte qu'il ait une qualité satisfaisante pour les mesures de fitness, précision, généralisation et simplicité. Étant donné qu'il y a un équilibre entre ces différentes mesures de qualité, les comportements peu fréquents observés dans le log ne sont pas forcément modélisés car il y a de fortes chances que leur impact positif sur la Fitness soit inférieur à leur impact négatif sur les autres

mesures. Enfin, les process trees sont capables de modéliser la quasi-totalité des comportements observables tels que les situations de choix ou de parallélisme, à l'exception des non-free choice constructs.

Chaque contrainte prise séparément, nous sommes tout à fait conscient qu'il existe des techniques qui conviennent mieux que l'Inductive Miner. Cependant, pour les raisons énnoncés ci-dessus, l'Inductive Miner est la technique qui satisfait le plus nos critères et sauf mention explicite, sera celle utilisée pour la génération des différents modèles présentés dans la suite de ce manuscrit. Dans la sous-section suivante, nous présentons brièvement le fonctionnement de l'Inductive Miner et définissons les process trees.

3.3.1.2 L'Inductive Miner

Traces
⟨ **A**,B,C,D,E,G ⟩
⟨ **A**,B,C,D,F,G ⟩
⟨ **A**,C,D,B,F,G ⟩
⟨ **A**,B,D,C,E,G ⟩
⟨ **A**,D,C,B,F,G ⟩

(a)

Traces
⟨ B,C,D,E,**G** ⟩
⟨ B,C,D,F,**G** ⟩
⟨ C,D,B,F,**G** ⟩
⟨ B,D,C,E,**G** ⟩
⟨ D,C,B,F,**G** ⟩

(b)

Traces
⟨ B,C,D,**E** ⟩
⟨ B,C,D,**F** ⟩
⟨ C,D,B,**F** ⟩
⟨ B,D,C,**E** ⟩
⟨ D,C,B,**F** ⟩

(c)

Traces
⟨ **B**,**C**,**D** ⟩
⟨ **B**,**C**,**D** ⟩
⟨ **C**,**D**,**B** ⟩
⟨ **B**,**D**,**C** ⟩
⟨ **D**,**C**,**B** ⟩

(d)

FIGURE 3.8 – Exemple de découpe d'un log en sous-logs par l'Inductive Miner.

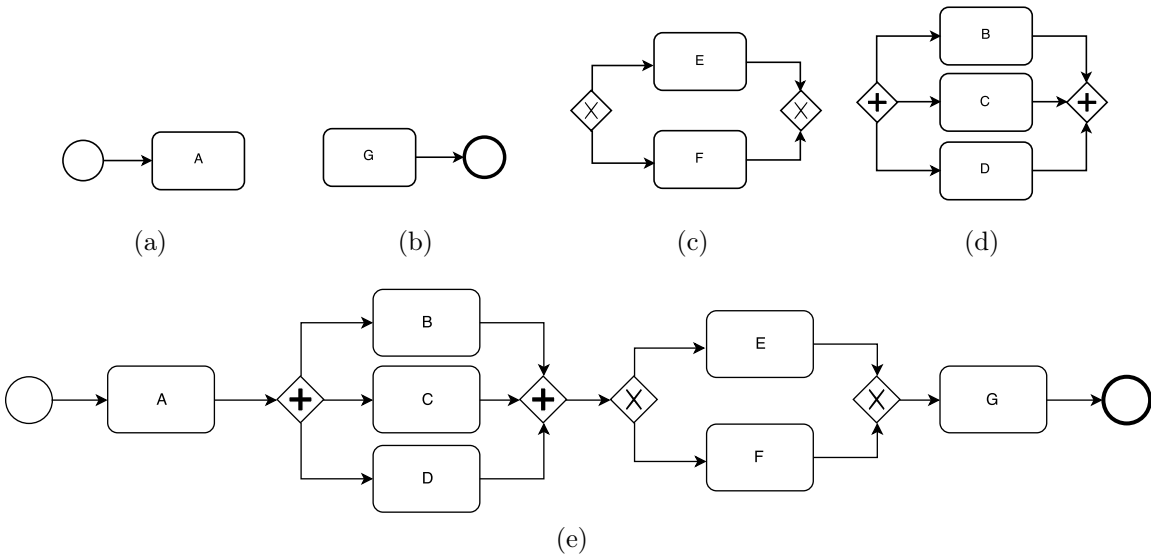


FIGURE 3.9 – Exemple de diagramme BPMN construit par l'Inductive Miner sur le log de la Figure 3.8a.

L'Inductive Miner est une technique dont le principe est d'effectuer une découpe "intelligente" de l'event log en sous-logs, et ce de façon itérative afin de construire un modèle final. Nous illustrons ce principe à partir de l'event log présenté dans la Figure 3.8a. A partir de ce log, plusieurs découpes sont faciles à remarquer. Tout d'abord, il est possible d'effectuer une découpe après le premier event de chaque trace. A partir du sous log de

gauche composé uniquement de l'activité "A", l'Inductive Miner construit le fragment de diagramme BPMN présenté dans la Figure 3.9a. Le sous log restant est présenté dans la figure 3.8b. La aussi, nous remarquons qu'une découpe sur le dernier event de chaque trace est possible, résultant au fragment de diagramme BPMN présenté dans la Figure 3.9b. A partir du sous-log restant de la Figure 3.8c, nous remarquons que chaque trace se termine soit par l'activité "E", soit par l'activité "F". Une découpe est donc possible et une situation de choix exclusif est construite par l'Inductive Miner, présentée dans la Figure 3.9c. Enfin, à partir du sous-log restant, nous remarquons que les activités "B", "C" et "D" sont toutes les trois exécutées mais dans un ordre arbitraire ; une situation de parallélisme est donc construite dans la Figure 3.9d. Une fois rassemblés, ces fragments composent le diagramme BPMN final, présenté dans la Figure 3.9e. Bien sûr, certains log ne permettent pas une découpe aussi évidente, cet exemple étant volontairement simplifié.

3.3.1.3 Les process trees

L'Inductive Miner n'utilise pas de modèle BPMN comme structure algorithmique, mais plutôt un process tree. En nous inspirant de [Buijs, 2014], nous définissons un process tree comme suit :

Définition 13 - Process Tree :
Soit $A \subseteq \mathcal{A}$ un ensemble fini d'activités. Un Process Tree PT est un tuple PT = (N, r, m, c) où :

- *N est l'ensemble (ordonné) non vide de nœuds dans le process tree, composé de deux sous-ensembles, N_L pour les nœuds-feuille et N_O pour les nœuds-opérateur tels que $N_L \cup N_O = N$ et $N_L \cap N_O = \emptyset$. Il est possible de faire référence à un nœud $n \in N$ en fonction de sa position $i \in \mathbb{R}^{+*}$ dans le process tree selon un parcours en ordre préfixé ; i.e. un parcours en profondeur qui va de la racine aux nœuds-feuille par les nœuds enfants les plus à gauche tant que cela est possible. Cette référence est notée n_i, n_1 étant la racine de l'arbre.*
- *$r \in N$ est le nœud-racine.*
  O est l'ensemble composé des différents types d'opérateur : $O = \{\rightarrow, \leftarrow, \times, \wedge, \vee, \circlearrowleft\}$
- *$m : N \rightarrow A \cup O$ est une fonction affectant chaque nœud à un opérateur ou une activité :*

$$m(n) = \begin{cases} a \in A \cup \{\tau\}, & si\ n \in N_L, \\ o \in O, & si\ n \in N_O. \end{cases} \tag{3.2}$$

- *$c : N \rightarrow N^*$ est la fonction de succession :*
  $c(n) = \langle\rangle$, si $n \in N_L$,
  $c(n) \in N^+$, si $n \in N_O$.

  tel que :
  - *chaque nœud, excepté le nœud-racine, a exactement un parent :*
    $\forall n \in N \backslash \{r\} : \exists! p \in N_O : n \in c(p)$
  - *le nœud-racine n'a pas de parent :*
    $\nexists n \in N : r \in c(n)$

— *chaque nœud apparaît seulement une fois dans la liste des enfants de son parent :*
$\forall n \in N : \forall 1 \leq i < j \leq |c(n)| : c(n)_i \neq c(n)_j$

— *un nœud-opérateur de type 'boucle' a exactement trois enfants :*
$\forall n \in N : (m(n) = \circlearrowleft) \Rightarrow |c(n)| = 3$

— $s : N \to N^*$ *est la fonction sous-arbre qui renvoie tous les nœuds de* n *dans l'ordre préfixé.*

$$s(n) = \begin{cases} n, & si\ n \in N_L, \\ n \cdot s(c(n)_1) \cdot \ldots \cdot s(c(n)_{|c(n)|}), & si\ n \in N_O. \end{cases} \tag{3.3}$$

Le sous-arbre d'un nœud $n \in N$ *peut être noté comme suit :* $n = m(n).s(n)$.

— *Un process tree ne contient pas de récursion :*
$\forall n \in N \backslash \{r\} : \exists p \in N_O : (n \in c(p)) \wedge (p \notin s(n))$

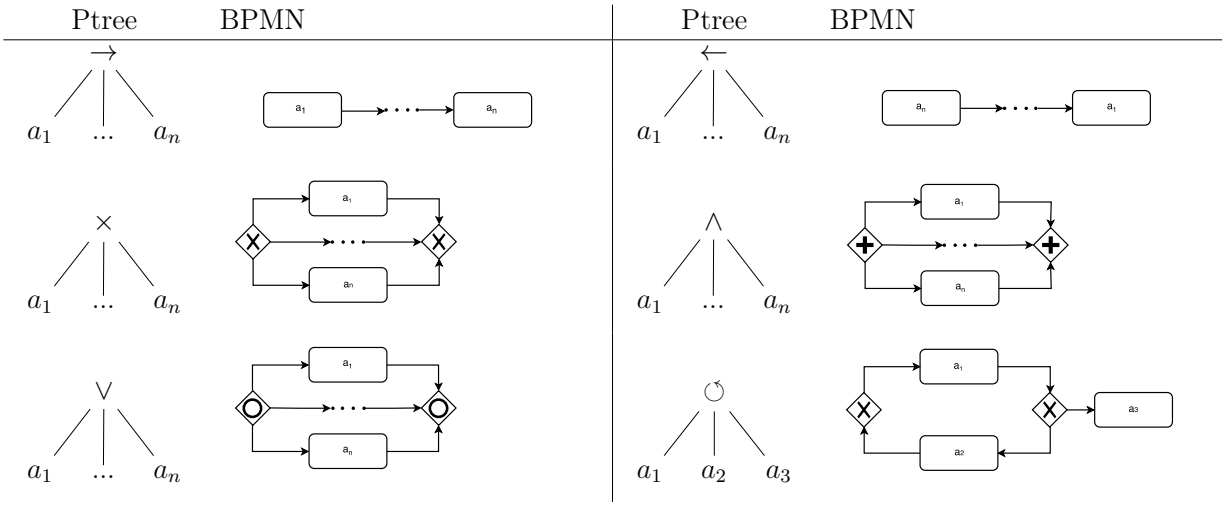


FIGURE 3.10 – Équivalence entre les notations process tree et BPMN.

Un exemple de process tree est présenté dans la Figure 3.11a. Sa notation préfixée est la suivante : $\rightarrow$ (n_1), A (n_2), $\times$ (n_3), $\leftarrow$ (n_4), C (n_5), B (n_6), D (n_7), $\wedge$ (n_8), E (n_9), F (n_{10}), $\vee$ (n_{11}), I (n_{12}), $\circlearrowleft$ (n_{13}), G (n_{14}), H (n_{15}), J (n_{16}), K (n_{17}).

Nous présentons dans la Figure 3.10 l'équivalence entre les différents opérateurs utilisés dans les process tree et les symboles utilisés dans les diagrammes BPMN. A l'aide de ces équivalences, nous pouvons traduire le process tree présenté dans la Figure 3.11a en diagramme BPMN, que nous présentons dans la Figure 3.11b.

Le nœud racine est un opérateur Séquence ($\rightarrow$), ce qui signifie que ses nœuds enfants devront être interprétés de gauche à droite. L'activité A est donc exécutée en premier, suivie d'un choix exclusif ; i.e., un seul choix est possible parmi les flux sortants ; représenté par l'opérateur XOR ($\times$) :

— le premier choix est l'exécution séquentielle des activités B et C en commençant par B. Cet ordre est représenté par l'opérateur Séquence inversée ($\leftarrow$).

— le deuxième choix est l'exécution simple de l'activité D.

— le troisième choix est l'exécution en parallèle des activités E et F. Cette exécution est représentée par l'opérateur AND ($\wedge$), signifiant que tous ses nœuds enfants devront être exécutés mais dans un ordre arbitraire.

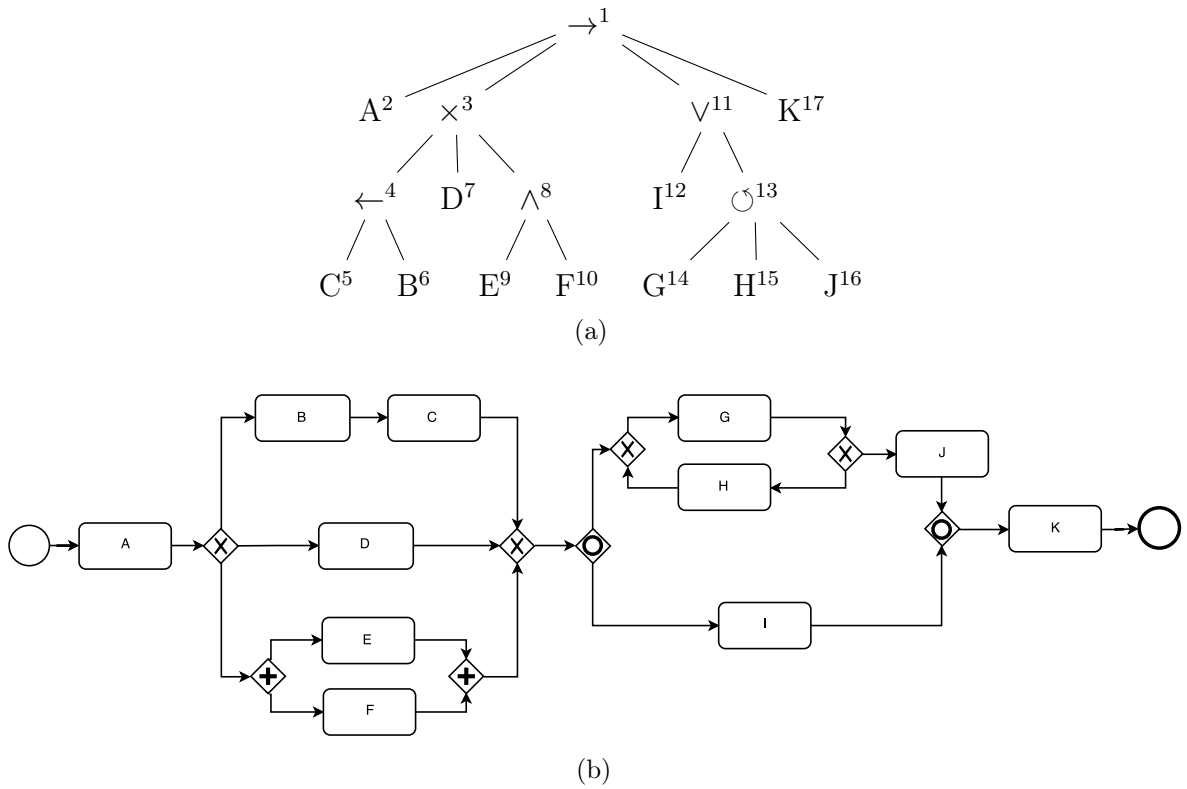


FIGURE 3.11 – Exemple d'un *(a)* process tree et *(b)* de son diagramme BPMN équivalent.

Le nœud suivant est un opérateur OR ($\vee$) et modélise un choix inclusif, signifiant qu'il est possible d'exécuter un ou plusieurs des flux sortants. Les choix possibles sont :

— l'exécution d'une boucle sur les activités G et H. D'abord l'activité G (le *do*) est exécutée. Si l'activité H (le *redo*) est exécutée après, alors l'activité G devra être exécutée de nouveau. Sinon, l'activité J (la sortie) est exécutée et nous sortons de la boucle. Le choix entre l'exécution de l'activité H et J est à faire tant que l'activité H a été exécutée au choix précédent et ce, pour un nombre infini de fois.

— l'exécution simple de l'activité I.

Enfin, l'activité K est exécutée et le processus se termine. Comme nous pouvons le voir avec cet exemple, la lecture d'un diagramme BPMN est assez simple.

Définition 14 - Langage d'un process tree :
Le langage d'un process tree, notée $\mathfrak{L}(PT)$, représente l'ensemble des exécutions possibles à partir de ce process tree.

3.3.2 Transformation de la structure algorithmique

Dans cette sous-section, nous instancions l'étape #2 de la méthodologie KITE : transformer la structure algorithmique. Comme nous l'avons défini dans la sous-section 3.2.1, l'objectif est de transformer la structure algorithmique manipulée lors de la construction du process tree pour (1) permettre un accès rapide à l'algorithme de fouille et (2) réduire l'espace de recherche.

Dans cette instanciation, garder un process tree pour détecter des relations entre activités n'est pas nécessaire. Nous avons juste besoin de savoir quelles activités sont

intéressantes à explorer. Par conséquent, un simple vecteur d'activités suffit. La question est de savoir quelles activités vont composer ce vecteur. Dans cette étape, nous nous attacherons à transformer le process tree construit en un vecteur d'activités .

La problématique que nous traitons est celle de l'impact des choix et de leur ordre dans l'exécution du processus. La structuration en arbre des process trees et la présence explicite des différents opérateurs permet d'avoir un accès direct aux comportements possibles, ce qui va permettre de définir quelles activités sont intéressantes à analyser, et donc quelle partie de l'espace de recherche il n'est pas utile d'explorer.

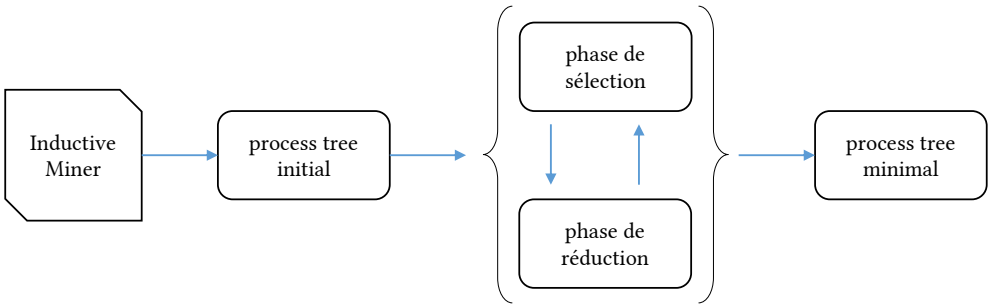


FIGURE 3.12 – Phases de sélection et de réduction pour transformer le process tree initial en process tree minimal.

Dans la suite de cette étape #2, notre contribution est l'élagage de l'espace de recherche pour détecter et évaluer les dépendances liées aux choix et aux ordres d'exécution des activités. Cet élagage va être effectué en analysant les blocs qui contiennent les différentes activités. Nous proposons les étapes de cet élagage dans la Figure 3.12. A partir d'un process tree initial construit par l'Inductive Miner, un cycle de sélection et de réduction est effectué afin de produire un arbre minimal. Nous détaillons les phases de sélection/réduction ainsi que le concept d'arbre minimal dans la suite de cette sous-section.

3.3.2.1 Phase de sélection

Dans un process tree, les différentes situations (choix, parralélisme, etc...) sont représentées par les différents nœuds-opérateur. Par conséquent, nous proposons de définir l'espace de recherche en retravaillant le process tree construit. C'est la phase de *sélection*. Avec un parcours en ordre préfixé, nous modifions le process tree à l'aide d'un ensemble de règles que nous définissons en fonction de la valeur de chaque opérateur rencontré :

— **Règle 1 (R1) : Séquence (→), Séquence inversée (←)** :
Les activités comprises dans des situations séquentielles partagent la caractéristique d'être obligatoirement exécutées, et ce dans un ordre précis. Autrement dit, leur exécution ne peut pas conditionner l'exécution d'autres activités. Pour ces deux opérateurs, nous considérons deux situations. La première est celle où tous les enfants d'un nœud-opérateur de type séquence ou séquence inversée sont des nœuds-feuille. Dans ce cas là, nous pouvons déduire l'exécution de n'importe quel enfant si l'on constate l'exécution du premier enfant. Par conséquent, nous pouvons ramener l'ensemble des enfants d'un nœud-opérateur de type Séquence ou Séquence inversée au premier enfant qui n'est pas une action silencieuse. La seconde situation est celle où au moins un des enfants d'un nœud-opérateur de type Séquence ou Séquence inversée n'est

pas un nœud-feuille. Dans ce cas là, il est possible de déduire l'exécution de l'un des nœuds-feuille à partir de l'exécution d'un nœud enfant de l'un des nœuds-opérateur. Par conséquent, nous pouvons ramener l'ensemble des enfants d'un nœud-opérateur de type Séquence ou Séquence inversée à l'ensemble des enfants nœuds-opérateur de ce nœud.
Si $m(n_i) \in \{\rightarrow, \leftarrow\}$, alors :

$$c(n_i) := \begin{cases} c(n_i)_k | \nexists p < k : c(n_i)_p \neq \tau, & \text{si } \forall n_j \in c(n_i) \cap N_L, \\ \bigcup_{j=1}^{|c(n_i)|} n_j | n_j \in c(n_i) \cap N_O, & \text{sinon.} \end{cases} \tag{3.4}$$

— **Règle 2 (R2) : Boucle (↻)** :
De manière générale, les boucles sont des situations compliquées à gérer. Les inclure dans notre problématique ne fait pas exception. La Figure 3.13 présente plusieurs boucles possibles sous la forme de process trees.

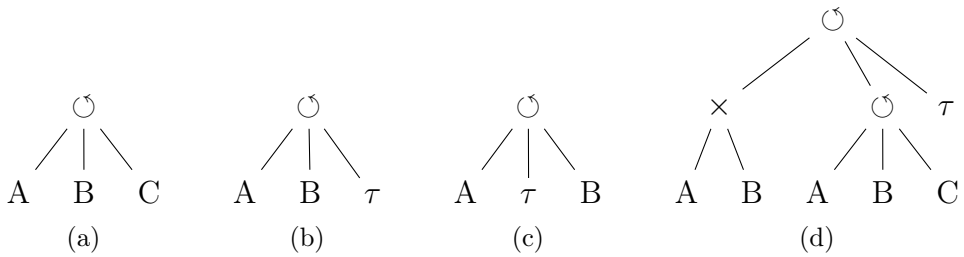


FIGURE 3.13 – Différentes situation possible comprenant des boucles.

La Figure 3.13a présente une boucle simple, où trois activités composent respectivement le do, le redo et la sortie de la boucle. Les Figures 3.13b et 3.13c présentent des boucles où soit le redo soit la sortie ne sont pas des activités à proprement parler mais sont représentées par une action silencieuse ; i.e., une action utilisée pour la validité du modèle mais qui n'apparaît pas dans le log. Enfin, la figure 3.13d présente une boucle complexe, où le do est composé d'un choix exclusif entre plusieurs activités et le redo est composé d'une autre boucle. Analyser les boucles peut être intéressant dans le sens où il peut y avoir une dépendance entre une activité précédemment exécutée et le nombre d'exécutions de la boucle. Pour savoir que la boucle a été exécutée, il faut constater au moins une exécution du do ou l'exécution de la sortie. Cependant, dans le cas où la sortie est une action silencieuse comme dans la figure 3.13b, rien n'apparaîtra dans le log. De plus, nous pouvons déduire l'exécution de la sortie à partir de l'exécution du do. Il est aussi possible de déduire le nombre d'exécutions du redo en retirant une occurrence au nombre d'exécutions du do. Lorsque la boucle est complexe, l'analyser peut vite devenir compliqué. Bien que certaines informations seraient intéressantes à extraire, il en resterait néanmoins une grande partie qui rendrait l'extraction plus longue et sans valeur ajoutée. Par conséquent, pour des raisons de simplification, nous ne considérerons dans PRISM que la partie *do* des boucles.
Si $m(n_i) =$↻, alors :

$$c(n_i) := c(n_i)_1 \tag{3.5}$$

— **Règle 3 (R3) : OR ($\vee$), XOR ($\times$)** :
Il n'y aura que peu de changements pour les nœuds-opérateur de type OR et XOR, représentant respectivement les choix inclusifs et exclusifs. En effet, les dépendances vont concerner principalement les activités en situation de choix. Pour ces deux opérateurs, nous considérons deux situations. La première est celle où le nœud-opérateur concerné n'est pas la racine de l'arbre. Dans ce cas là, tout enfant nœud-feuille est potentiellement concerné par une dépendance, et tout enfant nœud-opérateur peut lui aussi être composé d'activités concernées par une dépendance. Nous ne touchons donc à rien. La seconde situation est celle où la racine du process tree est un nœud-opérateur de type OR ou XOR. Dans ce cas là, aucun des enfants nœud-feuille ne peut être concerné par une dépendance. Par conséquent, nous pouvons ramener l'ensemble des enfants d'un nœud-opérateur de type OR ou XOR à l'ensemble des enfants nœuds-opérateur de ce nœud.
Si $m(n_i)\in(\vee, \times)$, alors :

$$c(n_i) := \begin{cases} \bigcup_{j=1}^{|c(n_i)|} n_j | n_j \in c(n_i) \cap N_O, & \text{si } n_i = r, \\ c(n_i), & \text{sinon.} \end{cases} \tag{3.6}$$

— **Règle 4 (R4) : AND ($\wedge$)** :
Les situations de parallélisme sont un cas particulier dans le cadre de notre problématique. Nous nous intéressons à la fois à l'impact des choix et à l'ordre de ces choix. Comme les activités à exécuter en parallèle peuvent l'être dans un ordre totalement arbitraire, il est nécessaire de conserver chacune de ces activités pour savoir si leur ordre est inclus dans une dépendance.

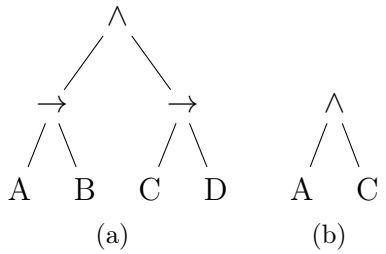


FIGURE 3.14 – (a) Situation de parallélisme et (b) une réduction respectant l'Équation 3.4.

Dans la Figure 3.14a, les deux enfants du nœud-opérateur AND sont des nœuds-opérateur de type Séquence. Le langage de ce process tree est $\mathfrak{L}(r) = \{\langle A, B, C, D\rangle, \langle A, C, B, D\rangle, \langle A, C, D, B\rangle, \langle C, D, A, B\rangle, \langle C, A, B, D\rangle, \langle C, A, D, B\rangle\}$.
Ces 6 séquences sont le nombre d'ordonnancements possibles de ces 4 activités. Si l'on respecte la réduction présentée par l'Équation 3.4, nous réduirions ce process tree à celui présenté dans la Figure 3.14b comportant seulement les activités A et C. Nous perdrions donc la possibilité de relever des dépendances relatives à la position des activités B et D dans cette situation de parallélisme.
Si $m(n_i)=\wedge$, alors :

$$s(n_i) := s(n_i) \tag{3.7}$$

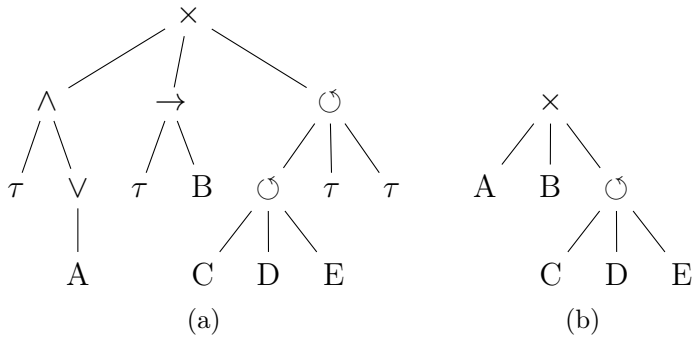


FIGURE 3.15 – (a) Process tree composé de nœuds inutiles et (b) sa version réduite.

3.3.2.2 Phase de réduction

Parcourir le process tree pour effectuer la phase de sélection n'est souvent pas suffisant. En effet, en modifiant la structure du process tree, il se peut que ce dernier se trouve dans un état *instable* à la fin de la phase de sélection. Un process tree est instable s'il contient des nœuds "inutiles" ; i.e., il est possible de réduire le process tree en supprimant des nœuds tout en assurant un langage identique. En nous inspirant de [Buijs, 2014], un nœud est inutile s'il respecte au moins une des conditions suivantes :

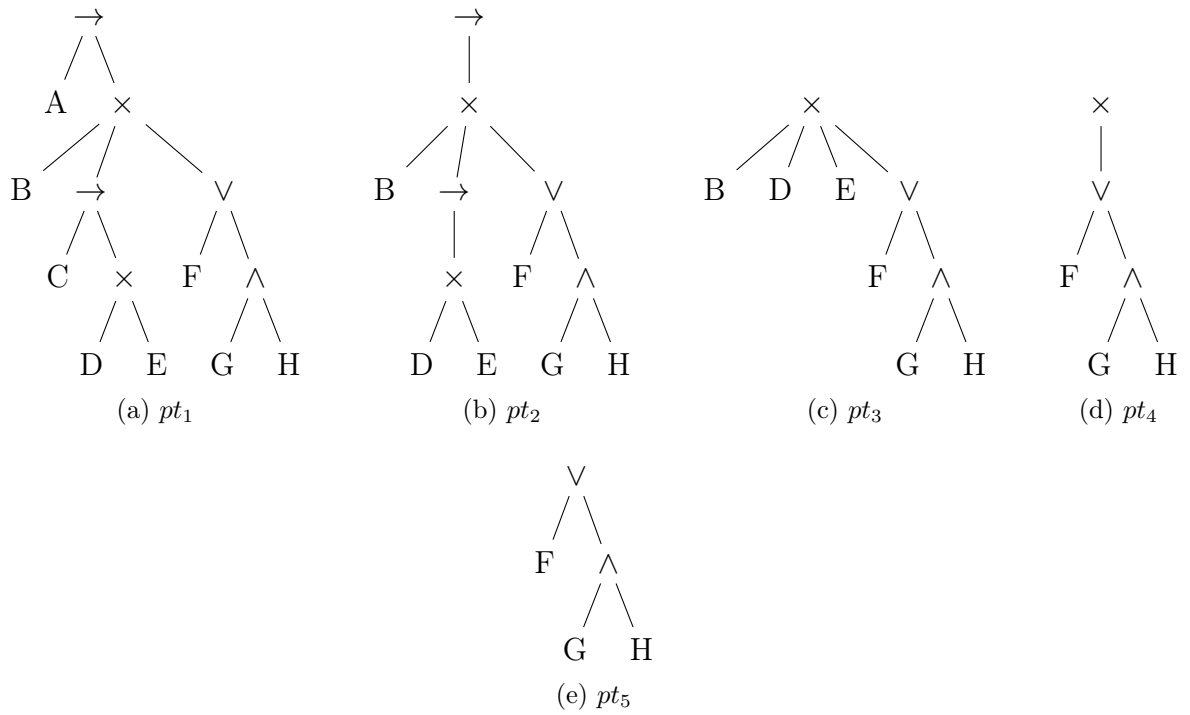
- **Règle 5 (R5) :** le nœud est une action silencieuse (τ) et n'est pas enfant d'un noeud-opérateur de type OR ou XOR.
- **Règle 6 (R6) :** le nœud est un nœud-opérateur avec un seul enfant.
- **Règle 7 (R7) :** le nœud est un nœud-opérateur dont tous les enfants sont des nœud inutiles.
- **Règle 8 (R8) :** le nœud est un nœud-opérateur Boucle dont le do est une boucle et dont le redo et la sortie sont des actions silencieuses (τ).
- **Règle 9 (R9) :** le nœud est un nœud-opérateur du même type que son parent (sauf si c'est un nœud-opérateur de type Boucle).

La Figure 3.15a présente un exemple de process tree comportant des nœuds inutiles. Le process tree présenté dans la Figure 3.15b est la version réduite, dont le langage reste identique au process tree original, à savoir $\mathfrak{L}(r)$: $\{\langle A\rangle, \langle B\rangle, \langle C, E\rangle, \langle C, D, C, E\rangle, \langle C, D, C, D, C, E\rangle, ...\}$ Ce langage est infini à cause de la boucle, dans laquelle les activités C et D peuvent être exécutées un nombre infini de fois.

Pour toutes ces situations possibles, il est nécessaire d'effectuer une phase de réduction afin de retrouver un process tree sans nœud inutile. Évidemment, le process tree réduit peut à nouveau être composé d'activités à ne pas analyser. Ainsi, un cycle de sélection/réduction est nécessaire jusqu'à ce que le process tree restant ne soit plus composé de nœuds inutiles et dont les activités restantes ne génèrent qu'un espace de recherche intéressant à explorer. Nous appelons ce process tree final le *process tree minimal*, et pour tout process tree $PT = (N, r, m, c)$, l'arbre minimal de PT est noté PT_{min}. L'Algorithme 1 présente cette phase itérative de sélection/réduction jusqu'à obtention du process tree minimal.

Algorithme 1 : Cycle de sélection/réduction

Entrées : PT=(N, r, m, c) : un process tree ;
tant que *PT n'est pas minimal* **faire**
  pour tous *nœuds* $n \in N_O$ **faire**
    appliquer la règle de sélection $Rs \in (R_1, R_2, R_3, R_4)$ sur n ;
  tant que *pt a des nœuds inutiles* **faire**
    pour tous *nœuds* $n \in N$ **faire**
      appliquer la règle de réduction $Rr \in (R_5, R_6, R_7, R_8, R_9)$ sur n ;



(a) pt_1 (b) pt_2 (c) pt_3 (d) pt_4 (e) pt_5

FIGURE 3.16 – (a) Process tree initial et (b), (c), (d), (e) un cycle de sélection/réduction pour obtenir un process tree minimal.

Un exemple de cycle de sélection/réduction est présenté dans la Figure 3.16. Notons pt_1, pt_2, pt_3, pt_4 et pt_5 les process trees présentés respectivement dans les Figures 3.16a, 3.16b, 3.16c, 3.16d et 3.16e.

1. pt_1 est le process tree initial. D'après la règle de sélection $R1$, les nœuds n_2 (A) et n_6 (C) de pt_1 peuvent être supprimés pour générer pt_2.
2. Ce nouveau process tree pt_2 contient des nœuds inutiles et une réduction est donc nécessaire. Par exemple, la règle de réduction $R6$ permet de supprimer $n_4(\rightarrow)$.
3. pt_3 est la version réduite de pt_2 mais contient à nouveau des nœuds qu'il est possible de supprimer. Par exemple, la règle de sélection $R3$ permet de supprimer n_2 (B), n_3 (D) ou encore n_4 (E).
4. Enfin, une dernière réduction sur pt_4 permet de générer pt_5, le process tree minimal, en supprimant n_1 $(\times)$ avec la règle de réduction $R6$.

Cela signifie que seules les relations entre les activités F G et H sont intéressantes à exploiter pour détecter des potentielles dépendances. Est-ce qu'exécuter l'activité F influe sur le fait d'exécuter ou non les activités G et H et/ou l'ordre dans lequel les exécuter ? Ce seront les questions auxquelles il sera possible de répondre sans avoir à explorer le reste de l'espace de recherche qui s'avère être sans valeur ajoutée.

Le vecteur d'activité V dont nous avons besoin pour définir si une activité doit être exploitée est simplement composé des activités représentées par les nœuds-feuille du process tree minimal. Pour un process tree minimal $PT_{min} = (N, r, m, c)$, V=$\{n \in N_L\}$.

C'est sur la production du process tree minimal et donc du vecteur d'activités, que s'achève l'étape #2 de PRISM, étape relative à la transformation de la structure algorithmique. Maintenant que nous avons réduit l'espace de recherche, il reste à l'explorer. Dans la sous-section suivante, nous détaillons l'étape #3 de PRISM, en introduisant *TWINCLE*, un algorithme que nous avons développé pour extraire des règles séquentielles à bas coût respectant des contraintes prédéfinies.

3.3.3 TWINCLE : proposition d'un algorithme pour la résolution du problème d'extraction de règles séquentielles à bas coût avec contraintes

Dans cette sous-section, nous allons effectuer l'étape #3 de la méthodologie KITE, étape centrale qui consiste à "fouiller" le log.

Comme nous l'avons défini dans la sous-section 3.2.1, l'objectif est (1) de détecter des dépendances potentielles (2) tout en considérant le contexte dans lequel elles prennent part.

Ce problème nous a conduit vers le problème d'extraction de motifs. D'une part, l'apprentissage non supervisé et les techniques d'extraction de motifs, et plus particulièrement d'extraction de *règles*, sont très similaires à notre problématique de détection de dépendances entre activités. On peut voir la corrélation indiquée par une règle comme une "dépendance" entre les activités dont elle est composée.

D'autre part, parmi les extensions qui ont été développées autour du problème d'extraction de motifs, certaines se concentrent particulièrement sur le concept d'utilité ; se référant ainsi à notre seconde problématique.

Après avoir introduit le problème d'extraction de motifs ainsi que ses concepts de base, nous nous concentrons sur l'extraction de motifs séquentiels, l'extraction de motifs profitables et l'extraction de motifs séquentiels profitables, extension récente de laquelle nous allons fortement nous inspirer dans nos travaux.

3.3.3.1 Extraction de motifs

L'extraction de motifs consiste à extraire des *motifs* intéressants et inattendus dans une base de données. Les motifs extraits peuvent être de plusieurs types : itemsets, règles d'association, sous-graphes, etc ...

Quel que soit le type de motif que l'on souhaite extraire, chaque technique se base sur un *dataset*. Le Tableau 3.1 présente un exemple de dataset issu d'une base de données transactionnelle. Nous supposons qu'une transaction représente les achats effectués par un client lors de son passage dans un supermarché. Ce dataset est composé d'un ensemble de *transactions* identifiées par un Id. Ces transactions sont composées de variables binaires,

Id	Café	Sucre	Biscuits
1	0	1	0
2	1	0	0
3	1	1	1
4	0	1	1

TABLEAU 3.1 – Exemple de base de données transactionnelle.

les *items*, indiquant si oui ou non l'achat est présent dans la liste. Un ensemble d'items est appelé *itemset* ; un *k-itemset* est un itemset composé de k items.

Quelques risques d'interprétation

Lorsqu'il s'agit d'expliquer un phénomène à l'aide de données, il est très facile de "mentir" volontairement ou non [Huff and Geis, 1954]. L'interprétation de résultats issus d'analyses de données est un exercice complexe comportant plusieurs risques.
"***Post hoc propter hoc***", "à la suite de cela, donc à cause de cela", est un sophisme qui consiste à prendre pour la cause ce qui n'est qu'un antécédent ; i.e., prétendre que si un événement B suit un événement A, alors le premier événement doit être la cause du second. La confusion entre ***corrélation*** et ***causalité***, aussi appelée ***effet cigogne*** (corrélation trompeuse entre le nombre de nids de cigognes et celui des naissances humaines), est particulièrement attirante parce que la séquence temporelle apparaît inhérente à la causalité.

Cependant, il faut être vigilant face à ces interprétations qui peuvent être fallacieuses. La corrélation peut être due à la **non-complétude des données** ("*on n'obtient que ce que l'on mesure*"). Tout ce qu'il peut se passer dans le processus réel n'a pas forcément été observé. Ce "manque" peut donc jouer dans la corrélation qui existe dans le log entre les différents éléments.
Quand bien même nous pourrions assurer la complétude du log, la corrélation de deux éléments A et B pourrait **avoir pour origine un troisième élément**, inconnu, non conscientisé ou non révélé.
Enfin, il n'est pas à exclure **la simple coïncidence**, il n'y a pas d'autre relation entre A et B à part qu'ils se sont produits l'un avant l'autre.

Dans la suite de ce manuscrit, nous essaierons dans la mesure du possible d'éviter toute ambiguïté dans les termes employés, afin de ne pas mener le lecteur vers une mauvaise compréhension des résultats présentés. Nous précisons que nos travaux ont pour but d'***observer*** et ***constater*** certains phénomènes existants. D'aucune manière nous nous risquons à interpréter ces résultats, exercice qui est du ressort d'experts des domaines étudiés.

Parmi les différentes techniques existantes, les plus connues sont l'*extraction d'itemsets fréquents* et l'*extraction de règles d'association*. L'extraction d'itemsets fréquents [Aggarwal and Han, 2014] vise à extraire les items ou ensembles d'items qui apparaissent souvent ensemble dans les différentes transactions. L'intérêt d'un itemset est évalué en fonction de son *support*. Le support d'un itemset mesure sa fréquence. C'est le ratio du nombre de transactions contenant cet itemset sur le nombre total de transactions dans le dataset. Par exemple, dans le dataset présenté dans la Figure 3.1, l'itemset {Sucre, Biscuits} ap-

paraît dans deux transactions sur quatre, son support est donc de 0,5. Cet itemset signifie donc que la moitié des clients ont acheté du sucre et des biscuits ensemble. L'objectif est d'extraire tous les itemsets dont le support est au dessus d'un support minimum prédéfini.

Quelques précisions lexicales

La littérature utilise différents termes pour désigner les concepts importants de l'extraction de connaissances, *item, itemset, pattern, etc....* Il arrive que certaines fois, par abus de langage ou par effets de traduction, ces termes soient utilisés indifféremment. Pour éviter toute confusion au lecteur, nous prenons le choix de définir un lexique de ces concepts qui sera utilisé dans ce manuscrit.
Le terme ***itemset*** sera traduit et utilisé tel quel. Le terme ***règle*** sera utilisé comme traduction du terme ***rule***. Nous utiliserons le terme ***motif*** comme traduction du terme ***pattern***. Un itemset et une règle seront considérés comme des ***types*** de motifs. Pour un itemset i, une règle r (ou plus généralement un motif m) donné, nous nommerons ***sur-ensemble*** tout ensemble E qui est au moins composé de i, de r (ou plus généralement de m). De la même façon, i, r (ou plus généralement m) sont appelés des ***sous-ensembles*** de E.

L'extraction de règles d'association [Agrawal et al., 1993] a pour objectif de découvrir des relations entre les variables de grandes bases de données. Une règle d'association est de la forme $X \rightarrow Y$, où X (appelé l'*antécédent*) et Y (appelé le *conséquent*) sont des itemsets. Tout comme l'extraction d'itemsets fréquents, le support est utilisé pour indiquer la fréquence de la règle. Le support d'une règle est égal au support de l'itemset composé de tous les items de la règle ; e.g., le support de la règle {Café} $\rightarrow$ {Sucre} est identique au support de l'itemset {Café, Sucre}. Pour la règle {Sucre} $\rightarrow$ {Biscuits}, le support est donc de 0,5. Cependant, une seconde mesure est utilisée pour évaluer la *robustesse* d'une règle : la *confiance*. La confiance d'une règle indique la proportion des transactions contenant l'antécédent de la règle qui contiennent aussi le conséquent. Pour une règle $X \rightarrow Y$, c'est le ratio du support de XY sur le support de X. L'itemset XY est l'itemset composé des items de X et des items de Y. Dans notre exemple, la règle {Sucre} $\rightarrow$ {Biscuits} a une confiance de 0,66 car sur les trois transactions qui contiennent du sucre, deux contiennent aussi des biscuits. Un itemset ou une règle est *fréquente* si son support est supérieur à un seuil prédéfini, et une règle est *valide* si sa confiance est supérieure à un seuil prédéfini.

Une approche naïve pour extraire tous les itemsets fréquents serait de tous les énumérer et d'évaluer leur support un à un. Sur la Figure 3.17 *(a)*, nous pouvons voir le treillis de l'ensemble des itemsets que nous aurions à étudier pour un dataset composé de quatre items A, B, C et D, soit 15 itemsets. A cause des problèmes "d'explosion combinatoire", cette approche n'est évidemment pas possible. Le nombre d'itemsets possibles pour un dataset de p items est égal à $\sum_{k=1}^{p} C_p^k$, ou plus simplement $2^p - 1$, soit environ 1,27 x 10^{30} pour un dataset de 100 items [Papon, 2016]. De la même façon, générer les règles valides revêt les même limites. Une méthode naïve peut être envisagée en deux phases : tout d'abord générer tous les itemsets fréquents, puis évaluer la confiance de l'ensemble des règles possibles à partir de chaque itemset fréquent. Pour chaque k-itemset pour k supérieur ou égal à 2, il existe $2^k - 2$ règles possibles, soit un total d'environ 5,15 x 10^{47} pour un dataset de 100 items ; $\sum_{k=2}^{p} C_p^k \times (2^k - 2)$.

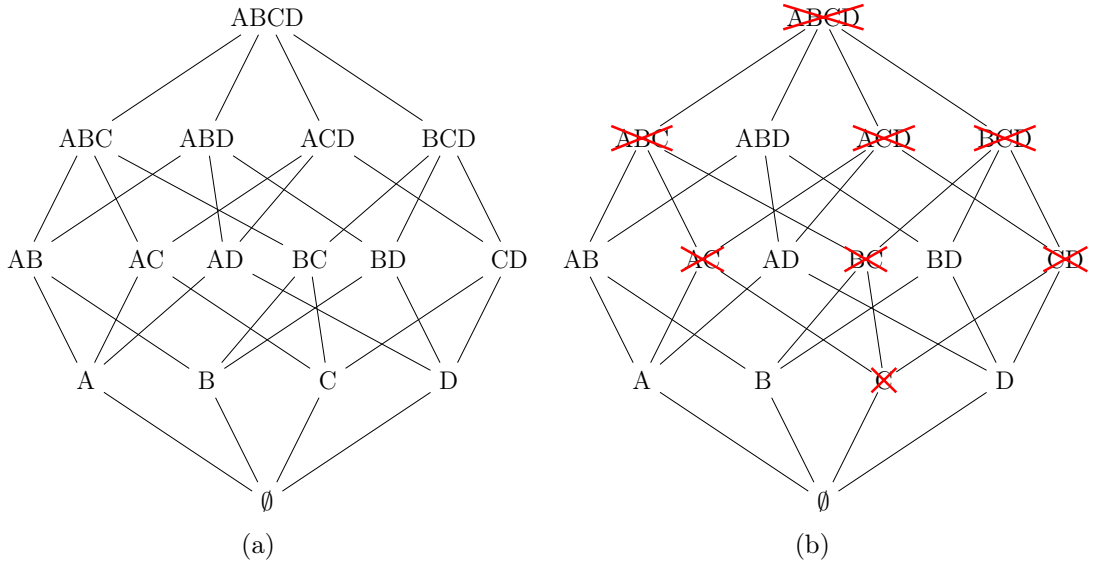


FIGURE 3.17 – *(a)* Le treillis de l'ensemble des itemsets possibles pour un dataset de quatre items A,B,C et D et *(b)* l'espace de recherche élagué depuis ce même treillis pour C non fréquent.

Pour pallier ces problèmes de temps d'exécution, de nombreux travaux se sont penchés sur des techniques pour réduire l'espace de recherche. Nous présentons l'algorithme de référence pour l'extraction de règles d'association : *Apriori* [Agrawal and Srikant, 1994]. Cet algorithme apporte des optimisations majeures à la fois dans la recherche des itemsets fréquents et dans la recherche des règles.

Pour la génération des itemsets fréquents, Apriori utilise la propriété d'*anti-monotonicité* du support pour réduire l'espace de recherche. Une mesure est anti-monotone si lorsqu'elle est vérifiée pour un itemset, elle est forcément vérifiée pour un sous-ensemble de cet itemset ; i.e., si l'itemset Y est fréquent, alors tout sous-ensemble X de Y sera fréquent. En d'autres termes, si un ensemble X n'est pas fréquent, alors tout sur-ensemble Y de X ne sera pas fréquent. Par conséquent, si le support d'un itemset X ne dépasse pas un support minimum prédéfini, alors aucun sur-ensemble Y de X ne dépassera ce support minimum. A l'inverse, une mesure est dite *monotone* si lorsqu'elle est vérifiée pour un itemset, elle est forcément vérifiée pour un sur-ensemble de cet itemset. Grâce à cette propriété anti-monotone du support, il est possible d'élaguer une partie de l'espace de recherche. Dans la Figure 3.17 *(b)*, nous présentons l'ensemble des itemsets que l'on peut élaguer si l'itemset C ne dépasse pas le support minimum. On remarque que l'on peut éviter ainsi l'évaluation de 7 itemsets {AC, BC, CD, ABC, ACD, BCD, ABCD}, soit presque la moitié. Pour utiliser cette propriété avec efficience, il faut donc générer les itemsets par ordre croissant de taille.

Pour la génération de règles, Apriori utilise une propriété de la confiance pour élaguer certaines règles sans avoir à calculer leur confiance. Considérons la règle $AB \rightarrow C$. Si cette règle ne respecte pas la confiance minimum, alors on sait que la règle $A \rightarrow BC$ ne le respectera pas non plus. En effet, pour les deux règles, le support est le même, soit le nombre de séquences dans laquelle apparaît l'itemset ABC. D'après la propriété anti-monotone du support, on sait que le support de A est supérieur ou égal à celui de AB. Donc si la confiance de $AB \rightarrow C$, soit le ratio du support de ABC sur le support de

AB ne respecte pas la confiance minimum, on sait que la confiance de la règle $A \rightarrow BC$, soit le ratio du support de ABC sur le support de A, ne le respectera pas non plus. Pour utiliser cette propriété de façon efficiente, pour chaque itemset fréquent, il faut commencer à évaluer les règles avec le conséquent le plus petit possible.

Apriori est l'algorithme de référence mais de nombreuses alternatives ont vu le jour. Eclat [Zaki, 2000] stocke pour chaque item la liste des transactions qui le contiennent dans un ensemble, le support des motifs étant calculé à partir de l'intersection de ces ensembles. FP-Growth [Han et al., 2004] utilise une structure nommée *FP-Tree* pour éviter la phase de génération des itemsets candidats. De plus, alors qu'Apriori fait un scan du dataset pour chaque ensemble d'itemsets de taille k, FP-Growth ne nécessite que deux passages, ce qui réduit considérablement les temps d'exécution.

D'autres algorithmes souhaitent extraire des itemsets plus "intéressants". Dans *FP-Close* [Grahne and Jianfei, 2005] et *FPMax* [Grahne and Jianfei, 2003], les auteurs cherchent à extraire respectivement les itemsets "*fermés*" (*closed*) et "*maximaux*" (*maximal*). Un itemset est dit "fermé" si aucun de ses sur-ensembles ne possède le même support que lui. Un itemset est dit "maximal" s'il est fréquent et qu'aucun de ses sur-ensembles n'est fréquent. Dans [Soulet and Rioult, 2014], les auteurs introduisent l'algorithme *DefMe* pour l'extraction d'itemsets "*générateurs*". Un itemset est dit "générateur" si aucun de ses sous-ensembles ne possède le même support que lui. Les itemsets fermés, maximaux et générateurs permettent l'extraction d'un nombre d'itemsets moins important mais ces itemsets sont reconnus être de meilleure qualité. Enfin, dans [Chang and Lee, 2003], l'algorithme *estDec* est introduit pour extraire les itemsets fréquents issus d'un flux continu de données (data stream), à l'inverse de la plupart des algorithmes qui se basent sur des datasets finis.

L'extraction des itemsets fréquents étant de loin la partie la plus coûteuse dans l'extraction de règles d'association, les techniques citées précédemment sont souvent étendues avec une seconde étape d'extraction de règles comme nous l'avons vu avec Apriori. Cependant, certains nouveaux algorithmes sont créés spécialement pour l'extraction de règles d'association. Par exemple, nous pouvons citer *TopKRule* [Fournier-Viger et al., 2012b]. Au lieu d'extraire les règles ayant un support et une confiance au dessus d'un certain seuil, TopKRule cherche à extraire le set contenant les k règles les plus importantes, en fonction de leur support. De plus, il n'utilise qu'une seule étape, et non plus deux comme les algorithmes inspirés d'Apriori. Utiliser deux étapes serait contre-performant pour l'extraction de top-k règles car il faudrait extraire dans la première étape tous les itemsets dont le support est supérieur à zéro, sélectionner les k itemsets avec le support le plus élevé et enfin extraire les règles ayant une confiance minimum. Pour pallier ce problème, TopKRule est un algorithme basé sur des *expansions récursives* des règles, inspirées des expansions utilisées dans les algorithmes tels que FP-Growth. Pour une règle, une expansion est le fait d'ajouter un item à l'antécédent ou au conséquent d'une règle. Une fois le set composé de k règles valides ; i.e., dont la confiance est au dessus du seuil minimum, la valeur du support le plus bas du set sert de seuil minimum utilisé pour élaguer l'espace de recherche.

Ce panel d'algorithmes est la preuve qu'il est intensément étudié pour son utilité dans de nombreux domaines d'application tels que les systèmes de recommandation, la bio-informatique ou encore les diagnostics médicaux.

Parmi les nombreuses extensions au problème d'extraction de motifs qui ont vu le jour, certaines se rapprochent particulièrement des problématiques traitées dans ce manuscrit.

Nous portons notre attention sur deux extensions en particulier, l'extraction de motifs séquentiels et l'extraction de motifs profitables. Nous pensons que leurs caractéristiques et leurs différents objectifs sont une piste de réponse à nos limites posées par les techniques de process mining. La sous-section suivante présente ces deux extensions afin de définir les avantages et inconvénients de chacune.

3.3.3.2 Extraction de motifs séquentiels

Le problème d'extraction de motifs séquentiels a été proposé par [Agrawal and Srikant, 1995] comme extension à l'extraction de motifs avec la prise en compte d'une notion importante, celle d'ordre entre les items. Cette extension a été appliquée à l'extraction d'itemsets mais aussi à l'extraction de règles, comme c'est souvent le cas pour les différentes extensions qui ont été développées au cours des années. Jusque là, les datasets étaient composés de transactions dans lesquelles cette notion n'existe pas. Dans l'exemple présenté dans le Tableau 3.1, une transaction représente une liste d'achats, tous effectués en même temps. On ne se soucie pas de savoir quand a été acheté le café par rapport au sucre. Dans d'autres datasets, issus de bases de données séquentielles, cette notion d'ordre tient une importance primordiale.

Id	Séquence
1	TV[1],Console[1],Jeu[5],Livre[12]
2	Console[1],TV[1],Jeu[1]
3	Jeu[5],Console[1],Clavier[1]
4	Console[1],Clavier[3]

TABLEAU 3.2 – Exemple de base de données séquentielle.

Un dataset séquentiel est composé de *séquences*. Une séquence est une liste de transactions ordonnées dans le temps. [Agrawal and Srikant, 1995] définit l'extraction de motifs séquentiels comme "*la découverte de tous les motifs séquentiels possédant un support minimum défini par l'utilisateur, où le support d'un motif est le nombre de séquences contenant ce motif*".

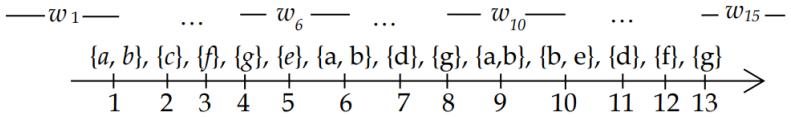


FIGURE 3.18 – Exemple de séquence temporelle.

Un exemple de dataset séquentiel est présenté dans le Tableau 3.2. Pour l'instant, faisons abstraction des chiffres entre crochets. Le dataset est composé de quatre séquences et nous supposons qu'une séquence correspond aux différents achats effectués par un client sur une période donnée. La première séquence nous indique qu'un client a d'abord acheté une TV et une console, ensuite un jeu, et enfin un livre. Une séquence dans laquelle chaque item est annoté d'un timestamp est appelée une *séquence temporelle* [Fournier-Viger et al., 2015]. Considérons le concept de "*fenêtre de temps*", représentant un groupe consécutif d'itemsets dans une séquence, et une "*fenêtre de temps glissante*", une fenêtre de temps qui se déplace du début à la fin de la séquence. Nous présentons un exemple de séquence temporelle dans la Figure 3.18, avec une fenêtre de temps glissante d'une longueur de

trois unités de temps. Il y a donc quinze fenêtres de temps différentes, notée w_i, sachant qu'une fenêtre de temps peut être en partie hors de la séquence pour que chaque itemset soit considéré un même nombre de fois.

Défini dans [Fournier-Viger et al., 2015], le problème d'extraction de règles séquentielles peut être considéré de trois façons différentes :

— ***Extraction de règles séquentielles dans une seule séquence :***
Connue principalement d'après les travaux de [Mannila et al., 1997] et l'algorithme *WINEPI*, une règle séquentielle dans une seule séquence est une règle telle que si les items de l'antécédent apparaissent, alors les items du conséquent vont aussi apparaître dans une certaine fenêtre de temps et avec une confiance donnée. L'utilisateur doit donc définir les seuils de support et de confiance minimum ainsi que la longueur de la fenêtre de temps. Le support d'une règle est le nombre de fenêtres de temps dans lesquelles apparaît la règle rapporté au nombre total de fenêtres de temps ; et la confiance est calculée en rapportant le support de la règle sur le support de l'antécédent de la règle. Dans la séquence présentée dans la Figure 3.18, le support de la règle $\{a,b\} \rightarrow \{g\}$ est de $1/15$ et la confiance est de $1/9$.

— ***Extraction de règles séquentielles entre séquences :***
Un exemple représentatif de ce problème est présenté par [Harms et al., 2002] avec l'algorithme *MOWCATL*. L'extraction de ce type de règles est sensiblement similaire à la précédente à l'exception que antécédent et conséquent n'ont pas l'obligation d'apparaître dans la même séquence. Par exemple, dans le dataset présenté dans la Table 3.2, la règle {Jeu, Console} → {Livre} apparaît une fois car l'antécédent apparaît dans la séquence #3 à la position 2 et le conséquent apparaît dans la séquence #1 à la position 4.

— ***Extraction de règles séquentielles communes à plusieurs séquences :***
Ce type de problème est le problème couramment traité et consiste à extraire les règles séquentielles pour lesquelles antécédent et conséquent apparaissent dans la même séquence et ce, pour un nombre minimum de séquences donné. Dans le dataset présenté dans la Table 3.2, la règle {Console} → {Jeu} a un support de $2/4$ et une confiance de $2/4$.

Quel que soit le problème traité, il existe aussi des différences dans le type de règles séquentielles que l'on veut extraire. Nous en distinguons deux : les "*règles séquentielles standard*", notées SSR pour "*standard sequential rules*", et les règles séquentielles partiellement ordonnées, notées POSR pour "*partially-ordered sequential rules*". Dans les deux types de règles, pour qu'une règle soit valide dans une séquence, tous les items de l'antécédent doivent apparaître avant tous les items du conséquent. Cependant, si l'on veut qu'une règle soit considérée comme une SSR, l'ordre des items à l'intérieur de l'antécédent et du conséquent doit aussi être pris en compte, sinon la règle est une POSR. Par exemple, dans le cas des POSR, les règles {Café, Sucre} → {Biscuit} et {Sucre, Café} → {Biscuit} sont confondues, alors qu'elles vont être considérées comme différentes dans le cas de SSR. De manière générale, les POSR ont prouvé donner de meilleurs résultats pour la prédiction [Fournier-Viger et al., 2012a].

Dans l'exemple du Tableau 3.2, considérons la règle séquentielle {Console} → {Jeu}. Son support et sa confiance valent tous deux 0,5. Cela nous indique que la moitié des clients

ont acheté une console puis un jeu, et que la moitié des clients ayant acheté une console ont plus tard acheté au moins un jeu. Évaluée comme règle d'association, le support et la confiance de cette règle auraient été de 0,75 à cause de la séquence #3 qui aurait été comptabilisée.

AprioriAll [Agrawal and Srikant, 1995] est le premier algorithme d'extraction de motifs séquentiels, qui a par la suite été optimisé avec la prise en compte de contraintes telles que des fenêtres de temps ou des taxonomies pour créer GSP [Srikant and Agrawal, 1996]. Les deux algorithmes sont largement inspirés d'Apriori. *Spade* [Zaki, 2001], inspiré d'Eclat, est une alternative à AprioriAll. Contrairement à AprioriAll qui utilise une *représentation horizontale* du dataset et une *recherche en largeur*, Spade utilise une *représentation verticale* et une *recherche en profondeur*. Un dataset horizontal est un dataset dans lequel chaque entrée représente une séquence d'itemsets, alors qu'un dataset vertical est un dataset dans lequel chaque entrée représente une liste de séquences dans lequel apparait un certain item. Une recherche en largeur va explorer et évaluer tous les motifs de taille k avant de passer aux motifs de taille k+1, alors qu'une recherche en profondeur va récursivement explorer, à partir d'un motif, tous les sur-ensembles de ce motif jusqu'à atteindre un seuil minimum et passer au motif suivant. Selon la construction de l'algorithme, une représentation/recherche sera plus adaptée qu'une autre. D'autres extensions telles que *SPAM* [Ayres et al., 2002] ou bitSPADE [Aseervatham et al., 2006]) utilisent les *bitvectors* pour calculer plus efficacement le support des itemsets. Pour un itemset donné, on affecte la valeur 1 au $i^{ème}$ bit de son bitvector si l'itemset apparaît dans la $i^{ème}$ séquence du dataset, et 0 s'il n'apparaît pas. *PrefixSpan* [Pei et al., 2004] est une autre alternative à AprioriAll dans laquelle la génération d'itemsets candidats à évaluer est remplacée par une analyse récursive du dataset pour ne créer que des itemsets existant à partir des itemsets fréquents. Enfin, à l'image de FPClose et FPMax, *CloSpan* [Yan et al., 2003] et *MaxSP* [Fournier-Viger et al., 2013] cherchent à extraire les itemsets séquentiels respectivement fermés et maximaux.

Pour l'extraction de règles séquentielles, *ERMiner* [Fournier-Viger et al., 2014] se base sur une représentation verticale pour éviter d'effectuer des projections du dataset. Il explore l'espace de recherche en utilisant le concept de "*classes équivalentes*" pour les règles ayant le même antécédent ou le même conséquent. De plus, il utilise une nouvelle structure, la *Sparse Count Matrix* pour élaguer l'espace de recherche. *TRuleGrowth* [Fournier-Viger et al., 2012c] est un algorithme pour extraire les règles séquentielles dont l'occurrence se situe dans une fenêtre de temps dont la longueur est prédéfinie. Les auteurs justifient cela par le fait que les utilisateurs ont souvent seulement besoin d'extraire des motifs apparaissant dans une certaine période. [Fournier-Viger and Tseng, 2013] proposent *TNS*, un algorithme pour extraire le set des top-k règles séquentielles non redondantes. Dans leurs travaux, les auteurs définissent qu'une règle r_1 est redondante vis-à-vis d'une règle r_2 si les deux règles possèdent le même support, la même confiance, que l'antécédent de r_2 est identique à celui de r_1 ou à l'un de ses sous-ensembles, et que le conséquent de r_1 est identique à celui de $r2$ ou à l'un de ses sous-ensembles.

Les règles séquentielles nous intéressent plus que les règles d'association dans la mesure où l'ordre des items a son importance dans notre problématique. Alors qu'une règle d'association pourrait indiquer s'il y a une corrélation dans l'apparition de deux activités, on ne pourrait pas savoir laquelle des deux semble être le "déclencheur" ; i.e., déterminer la "direction" de la dépendance.

Cependant, qu'elle soit avec ou sans contraintes, l'extraction de motifs séquentiels

conserve l'idée que la fréquence reste encore la mesure principale pour évaluer l'importance des différents motifs. Dans la sous-section suivante, nous allons introduire l'extraction de motifs profitables, une extension à l'extraction de motifs basée sur le concept de *profit*, duquel nous allons fortement nous inspirer dans nos travaux pour répondre à notre problématique qui consiste à évaluer les relations de dépendance en fonction de l'importance relative que nous attribuons aux activités concernées.

3.3.3.3 Extraction de motifs profitables

Le problème classique d'extraction de motifs fréquents se base sur un dataset où les quantités sont "binaires", soit l'item apparaît, soit il n'apparaît pas. Dans différents domaines d'application, cette condition n'a pas de sens. Par exemple, dans un dataset représentant des achats effectués, les clients ont pu acheter un ou plusieurs exemplaires d'un même produit. De la même manière, chaque produit n'a pas la même importance (différence de prix, de coût de production, etc...). Ne pas considérer ces quantités peut mener à l'extraction de motifs peu intéressants. De plus, certains motifs peuvent ne pas être fréquents mais représenter une grande valeur ; e.g., l'achat simultané de champagne et de caviar n'est pas fréquent mais reste générateur d'un grand profit. Pour pallier cette limitation, plusieurs extensions ont vu le jour. L'*extraction de motifs pondérés* [Yun, 2008, Chang, 2011] consiste à définir des "poids" à chaque item, indiquant leur importance relative. On considère que le poids d'un motif est la somme des poids de ce motif dans chaque séquence où il est présent. L'objectif est de découvrir tous les motifs ayant un poids au-dessus du poids minimum.

Item	TV	Console	Jeu	Clavier	Livre
Poids	100	70	40	30	20

TABLEAU 3.3 – Poids de chaque item dans l'intervalle [0,100].

L'*extraction de motifs profitables* est une extension à l'extraction de motifs pondérés. Ce problème, défini notamment dans [Chan et al., 2003] et [Yao et al., 2004], généralise l'importance relative des motifs en considérant, en plus des poids des items, leur quantité dans les séquences. L'association poids/quantité pour chaque item s'appelle le *profit*. De la même manière que pour le poids d'un motif, on considère que le profit d'un motif est la somme des profits de ce motif dans chaque séquence où il est présent.

Notons D le dataset que nous avons introduit dans le Tableau 3.2. Cette fois, nous allons nous servir des chiffres entre crochets, représentant la quantité achetée pour chaque item dans les différentes séquences ; e.g., Jeu[5] signifie que le client a acheté cinq jeux. Dans le Tableau 3.3, nous attribuons un poids compris dans un intervalle [0,100] à chaque item présent dans D. Par exemple, l'itemset {Jeu, TV} a un profit de (1x100 + 5x40) + (1x100 + 1x40) = 440.

Le problème d'extraction de motifs profitables est difficile car, contrairement au support, la mesure de profit ne possède pas de propriété monotone ou anti-monotone. Cela signifie que pour un itemset X ayant un profit p_1, un sur-ensemble Y de X peut avoir un profit p_2 inférieur, égal, ou supérieur à p_1. Donc il n'est pas possible d'utiliser le profit pour élaguer l'espace de recherche. Par conséquent, les différents algorithmes développés pour l'extraction de motifs profitables utilisent des bornes supérieures telle que la *Transaction Weighted Utilization* (TWU) qui elle est anti-monotone. La TWU d'un itemset X est égale à la somme des profits générés par chaque séquence contenant X. Par exemple

la TWU de l'itemset {Jeu, TV} est égal à (1x100 + 1x70 + 5x40 + 12x20) + (1x70 + 1x100 + 1x40) = 820. C'est donc une borne supérieure pour X et tout sur-ensemble Y de X ; i.e., ni l'itemset {Jeu, TV} ni aucun de ses sur-ensembles n'aura un profit supérieur à 820.

[Yao and Hamilton, 2006] ont proposé deux algorithmes, *UMining* et *UMining_H*, qui élaguent l'espace de recherche efficacement en appliquant une méthode d'estimation de l'utilité. Cependant cette stratégie possède l'inconvénient de pouvoir "rater" quelques itemsets. [Erwin et al., 2008] proposent *CTU-PROL*, un autre algorithme pour l'extraction d'itemsets profitables. Une structure arborescente appelée *Compressed Utility Pattern Tree* (CUP-Tree) est utilisée pour explorer l'ensemble des itemsets profitables. Si le dataset est trop grand pour être maintenu en mémoire, l'algorithme le divise à l'aide de projections parallèles pour chacune desquelles un CUP-Tree est utilisé. [Liu and Qu, 2012] proposent l'algorithme *HUI-Miner* ainsi qu'une nouvelle structure de donnée, l'*Utility-List*. Cette structure permet de limiter la génération de candidats et d'éviter de multiple parcours de la base de données. En contrepartie, le calcul du profit d'un itemset s'avère être une opération coûteuse. [Tseng et al., 2013] ont développé *UP-Growth*, un algorithme qui diminue efficacement la valeur de l'utilité estimée, réduisant ainsi le temps de calcul ainsi que le nombre d'itemsets extraits. Plus récemment, [Zida et al., 2017] introduisent *EFIM*. Cet algorithme se base sur deux bornes supérieures *sub-tree utility* et *local utility*, pour élaguer plus efficacement l'espace de recherche. De plus, il intègre une nouvelle technique appelée *Fast Utility Counting* pour calculer les bornes supérieures en temps linéaire.

Les algorithmes tels que *Two-Phase* [Lui et al., 2006], *IHUP* [Ahmed et al., 2009] ou *UP-Growth* utilisent une méthode en deux étapes. Dans la première, ils élaguent les itemsets non profitables à l'aide d'une borne supérieure telle que la TWU. Dans la seconde étape, ils évaluent réellement le profit de chaque itemset en parcourant le dataset et n'extraient que ceux au dessus d'un certain seuil. Cependant, ces deux étapes conservent beaucoup d'itemsets en mémoire. Pour pallier ce problème, d'autres algorithmes tels que *FHM* [Fournier-Viger et al., 2014], *HUI_Miner* ou *USpan* n'utilisent qu'une seule étape. Pour cela, ils introduisent le concept de "*profit restant*", qui représente pour un itemset X, la somme des profits de chaque item pouvant être ajouté à X. La principale borne supérieure utilisée par ces algorithmes pour élaguer l'espace de recherche est la somme du profit et du profit restant des différents itemsets.

Nous remarquons que de manière générale, l'objectif des algorithmes d'extraction de motifs profitables est de minimiser l'écart entre les bornes supérieures et les profits réels afin d'élaguer le plus grand espace de recherche possible.

Avec les applications pertinentes de l'extraction de motifs séquentiels et d'extraction de motifs profitables, l'extraction de motifs séquentiels profitables a été développée pour combiner les bénéfices des deux précédentes.

3.3.3.4 Extraction de motifs séquentiels profitables

L'extraction de motifs séquentiels profitables (EMSP) est motivée par le fait que l'utilité relative d'un motif est dépendante des items qui le composent mais aussi de l'ordre dans lequel ils apparaissent dans un motif. [Ahmed et al., 2010] proposent un framework pour étendre l'extraction de motifs séquentiels et introduisent deux algorithmes, *UL* et *US*. Dans leurs travaux, les auteurs calculent le profit selon si les séquences possèdent des occurrences distinctes ou non des items. L'algorithme *UMSP* est introduit dans [Shie et al., 2011] et utilise une structure nommée *MTS-Tree* pour extraire efficacement les

#	**Itemset**	**support**
1	Console	4
2	Jeu	3
	Console, Jeu	
3	TV	2
	Clavier	
	TV, Jeu	
	TV, Console	
	TV, Console, Jeu	
	Console, Clavier	
4	Livre	1
	TV, Livre	
	TV, Jeu, Livre	
	TV, Console, Livre	
	TV, Console, Jeu, Livre	
	Jeu, Livre	
	Console, Jeu, Livre	
	Console, Livre	
	Jeu, Clavier	
	Console, Jeu, Clavier	

(a) Itemsets Fréquents (EIF)

#	**Itemset**	**support**
1	Console	4
2	Jeu	3
3	TV	2
	TV, Jeu	
	Console, Jeu	
	Console, Clavier	
	Clavier	
4	TV, Console	1
	TV, Console, Jeu	
	TV, Console, Jeu, Livre	
	TV, Console, Livre	
	TV, Jeu, Livre	
	TV, Livre	
	Console, TV	
	Console, TV, Jeu	
	Console, Jeu, Livre	
	console, Livre	
	Jeu, Console	
	Jeu, Console, Clavier	
	Jeu, Clavier	
	Jeu, Livre	
	Livre	

(b) Itemsets Séquentiels (EIS)

#	**Itemset**	**profit**
1	Console, Jeu	650
2	TV, Console, Jeu, Livre	610
3	TV, Console, Jeu	580
4	TV, Jeu, Livre	540
5	Console, Jeu, Livre	510
6	Jeu, Livre	440
7	TV, Jeu	440
8	Jeu	440
9	TV, Console, Livre	410
10	TV, Livre	340
11	TV, Console	340
12	Console, Livre	310
13	Console, Jeu, Clavier	300
14	Console	280
15	Console, Clavier	260
16	Livre	240
17	Jeu, Clavier	230
18	TV	200
19	Clavier	120

(c) Itemsets Profitables (EIP)

#	**Itemset**	**profit**
1	TV, Console, Jeu, Livre	610
2	TV, Jeu, Livre	540
3	Console, Jeu, Livre	510
4	TV, Jeu	440
5	Jeu	440
6	Jeu, Livre	440
7	TV, Console, Livre	410
8	Console, Jeu	380
9	TV, Console, Jeu	370
10	TV, Livre	340
11	Console, Libres	310
12	Jeu, Console, Clavier	300
13	Console	280
14	Jeu, Console	270
15	Console, Clavier	260
16	Livre	240
17	Jeu, Clavier	230
18	Console, TV, Jeu	210
19	TV	200
20	TV, Console	170
21	Console, TV	170
22	Clavier	120

(d) Itemsets Séquentiels Profitables (EISP)

FIGURE 3.19 – Classement des itemsets pour chaque problème en fonction de leur mesure principale.

itemsets séquentiels profitables. Dans leurs travaux, [Ahmed et al., 2014] développent un algorithme spécifique à l'extraction d'itemsets séquentiels profitables depuis un dataset composé de logs de pages web. Là aussi, des structures arborescentes sont utilisées : le *UWAS-Tree* et le *IUWAS-Tree*. Introduit par [Yin et al., 2012], l'algorithme *USpan* utilise les *Lexicographic Quantitative Sequence Tree* (LQS-Tree) pour extraire le set complet d'itemsets séquentiels profitables. L'algorithme utilise des méchanismes de concaténations pour calculer le profit d'un itemset, ainsi que deux stratégies pour efficacement élaguer l'espace de recherche.

En 2015, [Zida et al., 2015] formalisent le problème d'extraction de règles séquentielles profitables. Selon les auteurs, ce problème diffère de celui d'extraction de motifs profitables dans la mesure où les algorithmes tels que *USpan*, *HUI_Miner* ou *FHM* utilisent des méthodes très différentes des algorithmes d'extraction de motifs séquentiels. De plus, la confiance doit aussi être calculée. Pour y parvenir, ils proposent l'algorithme *HUSRM*. Cet algorithme utilise une struture nommée *Utility-table* qui va permettre de calculer efficacement le support d'une règle. Cette structure va aussi évaluer pour chaque règle son profit (iutil), le profit restant pour tous les items pouvant être ajoutés à l'antécédent de la règle (lutil), le profit restant pour tous les items pouvant être ajoutés au conséquent de la règle (rutil), et le profit restant pour tous les items pouvant être ajoutés aussi bien à l'antécédent qu'au conséquent de la règle (lrutil). Avec l'utilisation de bitvector, *HUSRM* prouve être performant dans l'extraction de règles séquentielles profitables.

Le Tableau 3.4 est un récapitulatif des algorithmes présentés pour le problème d'extraction de motifs fréquents (EMF) et d'extraction de motifs séquentiels (EMS). Le Tableau 3.5 est un récapitulatif des algorithmes présentés pour le problème d'extraction de motifs profitables (EMP) et d'extraction de motifs séquentiels profitables (EMSP). La colonne problème indique à quel type de problème l'algorithme répond. Pour un problème donné, la colonne "Motifs" indique quel type de motifs l'algorithme s'attache à extraire. Même si certains algorithmes se concentrent sur l'extraction d'itemsets, nous avons attribué la valeur "tous" lorsqu'il est possible de les étendre avec une phase "standard" d'extraction de règles. Nous présentons dans la colonne "Caractéristiques" les quelques propriétés qui les distinguent les uns des autres. Nous précisons aussi dans cette colonne de quels algorithmes ils sont inspirés lorsque les auteurs le revendiquent clairement.

Afin de montrer que leurs différences impactent les motifs extraits, nous présentons dans la Figure 3.19 le classement des itemsets les plus importants selon :

— *(a)* le problème d'extraction d'itemsets fréquents (EIF)
— *(b)* le problème d'extraction d'itemsets séquentiels (EIS)
— *(c)* le problème d'extraction d'itemsets profitables (EIP)
— *(d)* le problème d'extraction d'itemsets séquentiels profitables (EISP)

Les classements sont faits en fonction de la mesure principale utilisée pour chaque problème ; i.e., le support pour l'EMF et l'EMS et le profit pour l'EIP et l'EISP. On notera 1, 2, ..., n le rang de l'itemset (ou du groupe d'itemsets en cas d'égalité de la valeur de la mesure utilisée) dans le classement.

Nous remarquons que les itemsets considérés comme importants changent selon le problème, chaque classement étant assez différent. Pour illustrer ces différences, nous allons prendre pour exemple deux itemsets : {Jeu, Console} et {TV, Jeu, Livre}.

Problème	Algorithme	Motifs	Caractéristiques
EMF	*Apriori*	tous	2 phases, génération de candidats
	Eclat	tous	dataset vertical
	FP-Growth	tous	nouvelle structure arborescente, expansions récursives
	FPClose	fermés	nouvelle structure arborescente, expansions récursives
	FPMax	maximaux	nouvelle structure arborescente, expansions récursives
	TopKRules	règles	extraction bornée, expansions récursives
	DefMe	générateurs	expansions récursives
	EstDec	tous	datasets non-finis (flux de données)
EMS	*WINEPI*	tous	considère des fenêtres de temps dans une seule séquence
	MOWCATL	règles	considère des règles réparties sur plusieurs séquences
	AprioriAll	tous	inspiré d'*Apriori*, 2 phases
	GSP	tous	inspiré d'*Apriori*, contraintes de temps et taxonomies
	Spade	tous	inspiré d'*Eclat*, dataset vertical
	SPAM	tous	utilisation de bitvectors
	bitSpade	tous	combinaison de *SPAM* et *Spade*
	PrefixSpan	tous	projection du dataset en plusieurs datasets plus petits
	CloSpan	fermés	inspiré de *PrefixSpan*
	MaxSP	fermés	inspiré de *PrefixSpan*
	ERMiner	règles	concept de classes équivalentes
	TRuleGrowth	règles	fenêtres de temps, expansions récursives
	TNS	règles	concept de non-redondance

Tableau 3.4 – Tableau récapitulatif des algorithmes présentés pour le problème d'extraction de motifs fréquents et le problème d'extraction de motifs séquentiels.

Pour les problèmes d'EIF et d'EIP, l'itemset {Jeu, Console} est confondu avec l'itemset {Console, Jeu}, l'ordre n'ayant pas d'importance. Au rang 2 pour l'EIF et au rang 1 pour l'EIP, cet itemset va donc être interprété, d'un point de vue de la fréquence d'apparition, comme un itemset très important. Cependant, si nous considérons les problèmes d'EIS et d'EISP, ce même itemset se retrouve dans la partie basse des classements. L'explication vient du fait que, les itemsets {Jeu, Console} et {Console, Jeu} étant distingués, le support de l'itemset {Jeu, Console} sera à priori [1] inférieur dans les problèmes d'EIS et d'EISP que dans les problèmes d'EIF et d'EIP. Cet itemset illustre donc l'impact de la prise en compte de l'ordre des items dans les algorithmes d'extraction de motifs.

Pour l'itemset {TV, Jeu, Livre}, ce n'est pas l'ordre qui va avoir son importance. En effet, si on regarde le classement des problèmes d'EIF et d'EIS d'une part, et des problèmes d'EIP et d'EISP d'autre part, on constate que le rang qu'il occupe est sensiblement le même. Nous pouvons l'expliquer justement par le fait que les itemsets avec lesquels il pourrait être confondu dans les algorithmes où l'ordre importe [2], n'apparaissent jamais. Cette fois, c'est l'importance relative attribuée qui va impacter les résultats. Si

1. Il sera égal si les itemsets avec lesquels il est confondu n'apparaissent jamais dans le dataset
2. {TV, Jeu, Livre}, {TV, Livre, Jeu}, {Jeu, TV, Livre}, {Jeu, Livre, TV}, {Livre, TV, Jeu}, {Livre, Jeu, TV}

Problème	Algorithme	Motifs	Caractéristiques
EMP	*Chan et al's*	tous	extraction top-k, inspiré d'*Apriori*
	Umining	tous	nouvelles méthodes d'élagage, inspiré d'*Apriori*
	Uminin_H	tous	méthode heuristique, inspiré d'*Umining*
	CTU-PROL	tous	projections parallèles, expansions récursives
	HUI-Miner	tous	nouvelle structure par listes
	UP-Growth	tous	nouvelle structure arborescente, réduction des bornes supérieures
	EFIM	tous	nouvelles bornes supérieures, projection du dataset
	Two-Phase	tous	2 phases, nouvelles bornes supérieures
	IHUP	tous	nouvelles structures arborescentes
EMSP	*FHM*	tous	nouvelles méthodes d'élagage
	UL	tous	2 phases, inspiré d'*Apriori*
	US	tous	expansions récursives
	UMSP	tous	nouvelles structures arborescentes
	Ahmed et al's	tous	nouvelles structures arborescentes
	USpan	tous	nouvelle structure arborescente, inspiré de *SPAM*
	HUSRM	règles	nouvelle structure par listes, nouvelles bornes supérieures

TABLEAU 3.5 – Tableau récapitulatif des algorithmes présentés pour le problème d'extraction de motifs profitables et le problème d'extraction de motifs séquentiels profitables.

l'on compare le classement des problèmes où le profit n'est pas pris en compte avec ceux où le profit est pris en compte, on constate que son rang dans le classement diffère. Cela s'explique par le fait que l'item TV est l'item avec le coût le plus élevé, et que les Livres et Jeux sont souvent vendus en plusieurs exemplaires. Le profit généré dépasse nettement l'itemset {Jeu, Console} qui occupe les premières places des classements dans lesquels le profit n'est pas pris en compte, car sa fréquence d'apparition ne contre-balance pas les faibles coûts et quantités vendues des items qui le composent. Cet itemset illustre donc l'impact de la prise en compte de l'importance relative des items dans les algorithmes d'extraction de motifs.

Nous voyons donc que selon l'objectif visé par chaque problème, les motifs extraits peuvent être très différents, ou en tout cas ne pas avoir le même rang dans le classement final. Nous allons nous servir de cette différence de perspective pour l'adapter à la modélisation de processus.

Nous présentons dans le Tableau 3.6 une grille de lecture récapitulative des algorithmes présentés, selon les critères que nous jugeons importants pour atteindre les différents objectifs listés dans ce manuscrit.

	Relations de dépendance					
Technique	bi./uni.	tot./fort.	dir./ind.	sim./mult.	ch./ord.	réf.
EMF		✓	✓	✓		fréquence
EMS	✓	✓	✓	✓	✓	fréquence
EMP		✓	✓	✓	✓	profit
EMSP	✓	✓	✓	✓	✓	profit

TABLEAU 3.6 – Récapitulatif des limites de l'existant dans les problèmes d'extraction de motifs.

Les critères de la grille concernent les relations de dépendance entre activités. Dans ce manuscrit, nous avons évoqué le besoin de détecter et différencier les relations bidirectionnelles (bi.), et unidirectionnelles (uni.), totales (tot.) et fortes (fort.), directes (dir.) et indirectes (ind.) simples (sim.) et multiples (mult.), de choix (ch.) et d'ordre (ord.). Les différents problèmes d'extraction de motifs fréquents (EMF), séquentiels (EMS), profitables (EMP) et séquentiels profitables (EMSP) peuvent détecter la plupart des relations de dépendance, à l'exception des relations unidirectionnelles qui ne peuvent être détectées que par les méthodes prenant en compte l'ordre des items (EMS et EMSP). Les problèmes d'EMP et d'EMSP ont l'avantage d'utiliser le référentiel (réf.) d'utilité qui nous intéresse plus pour traiter notre problématique, à l'inverse des problèmes d'EMF et d'EMS qui utilisent la fréquence.

En plus de cela, la prise en compte de l'ordre entre les items fait que les problèmes d'EMF et d'EMP génèrent des motifs de type "transaction", alors que les problèmes d'EMS et d'EMSP généreront des motifs de type "séquence". Dans nos travaux, l'ordre est un élément important, la génération à minima de séquences est donc un critère que nous considérons majeur. Enfin, chaque problème sans exception utilise une stratégie de maximisation ; i.e., les algorithmes cherchent à maximiser la mesure utilisée lors de l'extraction des motifs.

Nous remarquons donc que les techniques existantes en data mining possèdent certaines caractéristiques intéressantes mais restent limitées pour répondre à nos problématiques. Afin de pallier ces limites, nous proposons dans ce manuscrit le problème d'*Extraction de Règles Séquentielles à Bas Coûts* (ERSBC).

3.3.3.5 Formalisation du problème d'extraction de règles séquentielles à bas coût

Nous définissons le problème d'extraction de règles séquentielles à bas coût avec contraintes (ERSBC) [Dalmas et al., 2017a] comme une généralisation du problème d'extraction de règles séquentielles et comme une extension au problème d'extraction de règles séquentielles profitables. Les caractéristiques de ce nouveau problème sont les suivantes :

1. les algorithmes existants d'extraction de motifs profitables (EMP) sont, comme la plupart des algorithmes de fouille, basés sur une stratégie de maximisation de mesure ; i.e., extraire les motifs pour lesquels le profit est le plus haut possible. Comme son nom l'indique, le problème d'EMBC cherche à extraire les motifs avec le coût le plus bas possible. Nous avons donc besoin d'étendre le problème d'EMP avec une stratégie de minimisation de mesure. Dans cette extension, nous définissons le concept de *"coût"*, qui représente l'utilité relative d'un item. A l'inverse du profit, le coût est une valeur perçue plutôt comme une pénalité ; d'où la nécessité de minimiser cette mesure. Cette extension va donc conduire à l'introduction de nouveaux challenges.
2. Parmi les récents travaux qui traitent du problème d'extraction de motifs séquentiels, la majorité tendent à utiliser les *règles séquentielles partiellement ordonnées* (POSR) plutôt que les *règles séquentielles standard* (SSR). Les POSR sont un type spécifique de règles séquentielles dans lesquelles les items qui sont dans l'antécédent et le conséquent ne sont pas ordonnés, contrairement aux SSR dans lesquelles l'ordre compte à l'intérieur de l'antécédent et du conséquent [Fournier-Viger et al., 2015, Zida et al., 2015]. En plus de permettre d'obtenir de meilleurs résultats lorsqu'elles sont utilisées pour la prédiction, l'évaluation des POSR est souvent plus rapide

que celles des SSR. Cependant, dans nos travaux nous nous concentrons sur les SSR puisque nous accordons une importance capitale à l'ordre des items. Un ordre donné a une signification claire d'un point de vue médical.

3. le temps est un facteur clef quand il s'agit de traiter de parcours de soins [Konrad et al., 2010] et limiter la taille des itemsets permettrait d'extraire des motifs plus pertinents. Les travaux précédents ont utilisé de telles contraintes, mais à notre connaissance, aucun d'entre eux ne les a considérés ensemble au sein de SSR. De plus, nous introduisons un nouveau concept de *cohérence* pour augmenter la qualité des règles extraites.

Pour formaliser le problème d'extraction de règles séquentielles à bas coût, nous définissons tout d'abord différents concepts issus du problème classique d'extraction de motifs tout en les adaptant aux concepts déjà introduits en process mining. Nous introduisons par la suite les nouveaux concepts utiles à notre problème.

$\mathcal{L}$ \ t	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
σ_1	a		b		c			f	g	e
σ_2	a			d				c	b	e
σ_3	a		d			f	e			
σ_4	a		b	c	f	g				

TABLEAU 3.7 – Un exemple d'event log.

Définition 15 - Item, itemset, k-itemset :
Soit $I=\{i_1, i_2, ..., i_p\}$ *un ensemble de p items i. Un ensemble d'items est appelé un itemset, et nous disons qu'un itemset est un k-itemset s'il est composé de k items.*

Par convention, nous utilisons les majuscules pour représenter les itemsets et les minuscules pour représenter les items qu'ils contiennent.

Définition 16 - Règle séquentielle standard :
Soit A un ensemble fini d'activités. Une règle séquentielle standard (SSR) $r : X \rightarrow Y$ *(ou simplement r lorsque X et Y sont explicites) est une relation entre deux itemsets* $X=\{x_1, x_2, ..., x_n\} \in A$ *et* $Y=\{y_1, y_2, ..., y_m\} \in A$ *ordonnés tel que X et Y sont non vides* $X, Y \neq \emptyset$*, et que leur intersection est vide* $X \cap Y = \emptyset$.

Dans la suite de ce manuscrit, le mot "règle" fera référence à une SSR. On appelle l'itemset X l'antécédent de la règle et l'itemset Y le conséquent de la règle. Nous définissons l'itemset de la règle $\Omega(r)$ *comme l'union des itemsets X et Y,* $\Omega(r) = X \cup Y = \{X_1, X_2, ...X_{|X|}, Y_1, Y_2, ..., Y_{|Y|}\} = \{\omega_1, \omega_2, ..., \omega_{|X|+|Y|}\}$.
Soit une trace $\sigma = \langle \sigma(1), \sigma(2), ...\sigma(|\sigma|) \rangle$. *Dans le cas général, nous disons qu'un itemset I apparaît dans une trace* σ *(ou qu'une trace* σ *contient un itemset I), noté* $I \subseteq \sigma$*, si et seulement si tous les items de I apparaissent dans le bon ordre dans l'ensemble composé de la projection des activités de* σ*, soit* $I \subseteq \sigma \iff I \subseteq \bigcup_{i=1}^{|\sigma|} \pi_{activity}(\sigma(i))$ *et* $\forall\ i_p, i_q \in I$ *où* $\pi_{activity}(\sigma(k))=i_p$ *et* $\pi_{activity}(\sigma(l))=i_q$, $p < q \Rightarrow \sigma(k) \prec \sigma(l)$.
De la même manière, nous disons qu'une règle r apparaît dans une trace σ *(ou qu'une trace* σ *contient une règle r), noté* $r \subseteq \sigma$*, si et seulement si (1) tous les items de X et Y apparaissent dans* σ*, (2) que tous les items de Y apparaissent après tous les items de*

X dans σ et (3) que tous les items dans X et Y sont dans le même ordre dans σ, soit $r \subseteq \sigma \iff \Omega(r) \subseteq \sigma$.
Pour un event log $\mathcal{L}$, $tra(I)$ *et* $tra(r)$ *représentent respectivement l'ensemble des traces de* $\mathcal{L}$ *contenant l'itemset I et l'ensemble des traces de* $\mathcal{L}$ *contenant la règle r,* $tra(I)=\{\sigma|(\sigma\in\mathcal{L}) \wedge (I\subseteq\sigma)\}$ *;* $tra(r)=\{\sigma|(\sigma\in\mathcal{L}) \wedge (\Omega(r)\subseteq\sigma)\}$. *Pour des raisons de simplicité, nous définissons aussi* $ant(r)$ *l'ensemble des séquences contenant l'antécédent de la règle r,* $ant(r)=tra(X)=\{\sigma|\sigma\in\mathcal{L} \wedge X\subseteq\sigma\}$.

Nous illustrons ces définitions avec l'event log présenté dans le Tableau 3.7. Le log $\mathcal{L}$ est composé de 4 traces $\mathcal{L}=[\sigma_1, \sigma_2, \sigma_3, \sigma_4]$. Nous représentons le timestamp des différents events avec un indice $t \in \mathbb{R}^+$. Soit la règle $r : \{a, b\} \rightarrow \{c\}$. Les traces σ_1 et σ_4 contiennent r car {a,b,c} ⊆ {a,b,c,f,g} ⊆ {a,b,c,f,g,e} et que l'ordre entre les items est respecté. r n'est pas valide dans σ_2 car même si {a,b,c} ⊆ {a,d,c,b,e}, l'ordre entre b et c n'est pas respecté. Donc dans cet exemple, $tra(r)=\{\sigma_1, \sigma_4\}$. De la même façon $ant(r)=\{\sigma_1, \sigma_2, \sigma_4\}$ car seule σ_3 ne contient pas l'antécédent de r, soit {a,b}.

Définition 17 - Taille d'une règle :
*Une règle r : X→Y est de taille k*m si |X|=k et |Y|=m. Une règle de taille f*g est considérée plus grande qu'une règle de taille h*i si* $f > h$ *et* $g \geq i$, *ou que* $f \geq h$ *et* $g > i$.

En considérant l'event log présenté dans le Tableau 3.7, la règle $r : \{a, b\} \rightarrow \{c\}$ est de taille 2*1. La règle $t : \{a, b\} \rightarrow \{c, d\}$ est plus grande que r et la règle $q : \{a\} \rightarrow \{b\}$ est plus petite que r.

Définition 18 - Support :
Le support représente la fréquence de la règle ou la portée de la règle. Le support relatif (ou simplement support) d'un itemset I dans un event log $\mathcal{L}$ *est le rapport du nombre de traces dans* $\mathcal{L}$ *contenant I sur le nombre de traces total dans* $\mathcal{L}$, $sup_{\mathcal{L}}(I) = \frac{|tra(I)|}{|\mathcal{L}|}$.
Par extension, le support d'une règle r : X→Y dans un event log $\mathcal{L}$ *est le rapport du nombre de traces dans* $\mathcal{L}$ *contenant r sur le nombre de trace total dans* $\mathcal{L}$, $sup_{\mathcal{L}}(r) = \frac{|tra(r)|}{|\mathcal{L}|}$.

Il est à noter qu'il arrive de faire référence au support d'un itemset I (d'une règle r) dans un event log $\mathcal{L}$ simplement comme le nombre de traces dans $\mathcal{L}$ contenant I (r). Dans ce manuscrit, lorsque c'est la cas nous spécifions explicitement que nous faisons référence au support *absolu* pour éviter toute confusion avec la définition classique du support.

Définition 19 - Confiance :
La confiance représente la force de la règle. La confiance d'une règle r : X→Y dans un event log $\mathcal{L}$ *est le rapport du nombre de traces dans* $\mathcal{L}$ *contenant r sur le nombre de traces total dans* $\mathcal{L}$ *contenant l'antécédent de r,* $conf_{\mathcal{L}}(r) = \frac{|tra(r)|}{|ant(r)|}$.

En considérant l'event log présenté dans le Tableau 3.7, la règle $r : \{a, b\} \rightarrow \{c\}$ possède un support de 0,5 ; $sup_{\mathcal{L}}(r)=\frac{2}{4}=0,5$, ou un support absolu de 2. Sa confiance est d'environ 0,66 ; $conf_{\mathcal{L}}(r)=\frac{2}{3}\approx 0,66$.

Définition 20 - Problème d'extraction de règles séquentielles :
Soit minsup et minconf ∈ [0,1] représentant respectivement le support et la confiance minimum définis par un utilisateur, et $\mathcal{L}$ *un event log. Une règle r est fréquente si et seulement si son support est supérieur ou égal au support minimum,* $sup_{\mathcal{L}}(r) \geq minsup$. *Une règle est valide si et seulement si sa confiance est supérieure ou égale à la confiance minimum,* $conf_{\mathcal{L}}(r) \geq minconf$. *Le problème d'extraction de règles séquentielles depuis un event log consiste à extraire toutes les règles séquentielles fréquentes et valides.*

Quelques précisions

Les caractéristiques d'un event log que nous avons données dans la définition 11 sont celles d'un event log utilisé pour construire un modèle. Dans cette première partie de manuscrit, nous faisons une différence entre cet event log là et l'event log utilisé pour appliquer l'algorithme de fouille au niveau des occurrences d'une même activité dans une trace. Parce que nous voulons distinguer explicitement les différentes occurrences d'une activité dans une trace donnée, nous modifions le label des différentes occurrences. Pour une activité a, nous nommons $\#i\_a$ la i^{e} occurrence de l'activité a, $i \in \mathbb{N}^{+}$. Ainsi, une trace $\sigma = \langle a, b, c, a, b\rangle$ utilisée pour la construction d'un modèle deviendra $\sigma = \langle \#1\_a, \#1\_b, \#1\_c, \#2\_a, \#2\_b\rangle$ lorsque qu'elle sera utilisée pour la fouille. De ce fait, un élément du vecteur construit à la fin de l'étape $\#2$ fait référence à une même activité mais autorise la fouille de toutes les occurrences de cette activité.
Par exemple pour un vecteur V={a,b}, il sera possible de générer depuis la trace $\sigma = \langle \#1\_a, \#1\_b, \#1\_c, \#2\_a, \#2\_b\rangle$ les règles $\#1\_a \rightarrow \#1\_b$, $\#1\_a \rightarrow \#2\_a$, $\#1\_a \rightarrow \#2\_b$, $\#1\_b \rightarrow \#2\_a$, $\#1\_b \rightarrow \#2\_b$, $\#2\_a \rightarrow \#2\_b$.

Dans la même optique que l'algorithme *TRuleGrowth* [Fournier-Viger et al., 2015], nous pensons intéressant de définir une fenêtre de temps. Dans un domaine comme celui de la santé, il peut être pertinent de considérer que, si l'exécution d'un ensemble d'events est espacée ou non dans le temps, la relation qu'il existe entre ces events peut être interprétée différemment. Par exemple, la signification médicale de la succession des activités *crise* et *opération* à une heure d'intervalle peut être complètement différente de la succession de ces deux même activités à 15 heures d'intervalle. Par conséquent, nous définissons une fenêtre de temps dans laquelle une règle doit se trouver pour être valide.

Définition 21 - Fenêtre de temps :
Soit une règle $r : X \rightarrow Y$ *avec* $X=\{x_1, x_2, ..., x_n\}$ *et* $Y=\{y_1, y_2, ..., y_m\}$, *et une trace* σ *extraite d'un log* $\mathcal{L}$. *Nous définissons* $t_{beg}(r, \sigma)$ *comme le timestamp de l'event dont l'activité est représentée par le premier item de l'antécédent de* r, $t_{beg}(r, \sigma)=\pi_{time}(\sigma(i))$ *où* $\sigma \in \mathcal{L}$ *et* $\pi_{activity}(\sigma(i)) = x_1$. *De la même manière, nous définissons* $t_{end}(r, \sigma)$ *comme le timestamp de l'event dont l'activité est représentée par le dernier item du conséquent de* r, $t_{end}(r, \sigma)=\pi_{time}(\sigma(i))$ *où* $\sigma \in \mathcal{L}$ *et* $\pi_{activity}(\sigma(i)) = y_m$.
Nous définissons $tw(r, \sigma)$ *la durée d'une règle* r *dans une trace* σ *comme la durée qui sépare l'occurrence de l'event dont l'activité est représentée par le premier item de l'antécédent de* r *et l'occurrence de l'event dont l'activité est représentée par le dernier item du conséquent de* r, $tw(r, \sigma)=t_{end}(r, \sigma)\text{-}t_{beg}(r, \sigma)$.

De la même manière, nous définissons la durée d'une règle r *dans un log* $\mathcal{L}$ *comme la durée moyenne de* r *dans chaque trace* $\sigma \in \mathcal{L}$ *qui la contient,* $tw(r, \mathcal{L})=\frac{\sum_{\sigma \in tra(r)} tw(r,\sigma)}{|tra(r)|}$.
Nous notons tw_{max} *la fenêtre de temps maximum dans laquelle une règle doit se trouver dans une trace* σ *pour être retenue,* $tw(r, \sigma) \leq tw_{max}$.

En considérant l'event log présenté dans le Tableau 3.7, la règle $r : \{a, b\} \rightarrow \{c\}$ a une durée de 4 dans σ_1, $tw(r, \sigma_1)=5-1=4$; et de 3 dans σ_4, $tw(r, \sigma_4)=4-1=3$.

Dans la même optique que le problème d'extraction de motifs profitables, nous remettons en question l'utilisation de la mesure de support pour l'extraction de motifs

a	b	c	d	e	f	g
1	3	2	1	2	3	1

TABLEAU 3.8 – Table avec le coût par activité.

intéressants. Nous voulons à la place considérer la *valeur* des events dans l'évaluation des règles. Nous adaptons donc le concept de *profit* (1) aux events d'un event log et (2) en utilisant ces valeurs comme des *coûts* (des *pénalités*) plutôt que comme des profits. Pour cela, nous considérons dorénavant que chaque event présent dans le log possède un attribut *coût* ; i.e. il existe une fonction $\pi_{cost} : \mathcal{E} \to \mathbb{R}^+$ qui assigne un coût à chaque event du log. Nous définissons le *coût* d'une règle comme suit.

Définition 22 - Coût :
Le coût d'une règle r dans une trace $\sigma \in \mathcal{L}$, noté $c(r, \sigma)$, est la somme des coûts de chaque event de σ dont les activités sont représentées par les items de l'antécédent et du conséquent de r, $c(r,\sigma)= \sum_{\omega \in \Omega(r)} \sum_{i \in A_\omega} \pi_{cost}(\sigma(i))$ *avec* $A_\omega = \{j = 1, |\sigma| \ tq \ \pi_{activity}(\sigma_j) = \omega\}$.
De la même manière, nous définissons le coût d'une règle r dans un log $\mathcal{L}$ comme le coût moyen de r dans chaque trace $\sigma \in \mathcal{L}$ qui la contient, $c(r,\mathcal{L})=\frac{\sum_{\sigma \in tra(r)} c(r,\sigma)}{|tra(r)|}$.

Nous présentons dans le Tableau 3.8 un tableau de coût, qui affecte à chaque event un coût en fonction de l'activité qu'il contient. En considérant l'event log présenté dans le Tableau 3.7, la règle $r : \{a, b\} \to \{c\}$ a un coût de 6 dans σ_1 et σ_4, $c(r, \sigma_1)$=$c(r, \sigma_4)$=$1 + 3 + 2$=6. Dans notre exemple, comme le coût d'un event est basé sur son activité, il est normal que le coût d'une règle soit identique peu importe la trace qui la contient.

Contrairement aux algorithmes d'extraction de motifs profitables, nous décidons de prendre également en compte le temps d'exécution de la règle dans son coût. Cette nouveauté est motivée par les challenges posés par les domaines comme celui de la santé où il est crucial de minimiser à la fois le coût et le temps. Nous définissons donc le *coût global* d'une règle comme suit.

Définition 23 - Coût global :
Le coût global d'une règle r dans une trace $\sigma \in \mathcal{L}$, noté $oc(r, \sigma)$ est la considération du coût et de la durée d'exécution de r dans σ. Bien évidemment, les mesures de coût et de temps ne partagent pas les même unités et celles-ci ne sont ni prédéfinies ni même imposées. Par conséquent, il est nécessaire de définir k, un coefficient paramétré qui va être utilisé pour pondérer l'importance de chaque unité de coût par rapport à chaque unité de temps ; e.g. k=3 indique qu'une unité de coût est trois fois plus importante qu'une unité de temps. De cette façon, nous définissons $oc(r,\sigma)$=$c(r,\sigma) + k \cdot tw(r,\sigma)$.
De la même manière, nous définissons le coût global d'une règle r dans un log $\mathcal{L}$ comme le coût global moyen de r dans chaque trace $\sigma \in \mathcal{L}$ qui la contient, $oc(r,\mathcal{L})=\frac{\sum_{\sigma \in tra(r)} oc(r,\sigma)}{|tra(r)|}$.
Nous notons oc_{max} le coût global maximum qu'une règle peut avoir dans un log $\mathcal{L}$ pour être considérée valide, $oc(r,\mathcal{L}) \leq oc_{max}$.

En considérant l'event log présenté dans le Tableau 3.7, la table de coût présentée dans le Tableau 3.8 et k=2, la règle $r : \{a, b\} \to \{c\}$ a un coût global de 14 dans σ_1, $oc(r, \sigma_1)$=$6 + 2 \cdot 4$=14 ; et un coût global de 12 dans σ_4, $oc(r, \sigma_4)$=$6 + 2 \cdot 3$=12. Le coût global de r dans le log est de 13, $oc(r,\mathcal{L})=\frac{12+14}{2} = 13$.

Par l'extraction de règles séquentielles à bas coût, nous voulons permettre aux experts de santé d'avoir de nouvelles connaissances pertinentes sur les segments de parcours les moins coûteux. Pour qu'une règle soit pertinente, il faut donc qu'elle soit "régulière" en termes de coût et de durée dans les différentes traces qui la contiennent. Pour respecter cette contrainte de régularité, nous proposons une mesure de *cohérence*, que nous définissons comme suit.

Définition 24 - Cohérence :
La cohérence d'une règle dans un log mesure le degré de variation des différents facteurs comptabilisés dans le coût global de cette règle, soit le coût et la durée dans notre cas. Pour mesurer cette variation, nous utilisons le coefficient de variation qui est égal au rapport de l'écart-type sur la moyenne.
La cohérence d'une règle est donc définie comme la pondération du coefficient de variation de chaque facteur comptabilisé dans le coût global, $coh(r,\mathcal{L})=\alpha\cdot cv_{cost}(r,\mathcal{L})+\beta\cdot cv_{time}(r,\mathcal{L})$, *avec* $cv_{cost}(r,\mathcal{L}) = \frac{\sqrt{\frac{\sum_{\sigma\in tra(r)} |c(r,\sigma)-c(r,\mathcal{L})|}{|tra(r)|}}}{c(r,\mathcal{L})}$, $cv_{time}(r,\mathcal{L}) = \frac{\sqrt{\frac{\sum_{\sigma\in tra(r)} |tw(r,\sigma)-\mu^t|}{|tra(r)|}}}{\mu^t}$, $\mu^t = \frac{\sum_{\sigma\in tra(r)} c(r,\sigma)}{|tra(r)|}$ *et* $\alpha+\beta=1$.

En considérant l'event log présenté dans le Tableau 3.7 et en accordant la même importance aux différents facteurs pris en compte dans le calcul du coût global, $\alpha=\beta=0.5$, la règle $r : \{a,b\} \rightarrow \{c\}$ a une cohérence d'environ 0.20, $coh(r,\mathcal{L})=0,5\cdot\frac{0}{6}+0,5\cdot\frac{0.7}{3.5}\approx 0.20$.

En cherchant à extraire des règles à bas coût, TWINCLE va tendre à n'extraire que des règles de petite taille, qui risque de ne pas être intéressantes. Pour assurer un certain intérêt dans les règles extraites, il faut contraindre la taille des règles.

Définition 25 - Longueur :
Nous définissons μX, μY *et* $\mu\Omega \in \mathbb{R}^+$, *trois seuils de longueur. Pour qu'une règle* r *de taille k*m soit valide, il faut que* $k \geq \mu X$, $m \geq \mu Y$ *et* $k+m \geq \mu\Omega$, *avec* $\mu\Omega \geq \mu X + \mu Y$.

Entre contraintes classiques et nouvelles contraintes, nous définissons le problème d'extraction de règles séquentielles à bas coût avec contraintes comme suit.

Définition 26 - Problème d'extraction de règles séquentielles à bas coût avec contraintes :
Soit minconf $\in$ *[0,1],* tw_{max}, oc_{max}, μX, μY *et* $\mu\Omega \in \mathbb{R}^+$ *des seuils définis par l'utilisateur, et* $\mathcal{L}$ *un event log. Une règle est une règle à bas coût si* r *est valide et si son coût est inférieur au seuil de coût maximum. Le problème d'extraction de règles séquentielles à bas coût avec contraintes consiste à extraire toutes les règles à bas coût qui respectent les contraintes prédéfinies ; i.e., la fenêtre de temps et la longueur dans notre cas.*

Le problème d'extraction de règles séquentielles à bas coût est globalement difficile à résoudre. Chercher à minimiser le coût s'oppose aux techniques classiques qui cherchent à maximiser le support à deux niveaux :

— la mesure de coût global, à l'instar de celle de profit, n'est ni monotone ni anti-monotone ; i.e., pour deux règles r et t avec $r \subseteq t$, le coût global de t peut être supérieur, égal ou inférieur à celui de r. Considérons l'event log $\mathcal{L}$ présenté dans le Tableau 3.7, la table de coûts présentée dans le Tableau 3.8 et les règles $r : \{a\} \rightarrow \{b\}$, $t : \{a\} \rightarrow \{b,c\}$ et $q : \{a\} \rightarrow \{b,e\}$ avec $r \subset t$ et $r \subset q$. Pour k=6, le coût global de chacune des règles est :

— $oc(r, \mathcal{L})=\frac{(4+12)+(4+48)+(4+12)}{3}=28$
— $oc(t, \mathcal{L})=\frac{(6+24)+(6+18)}{2}=27$
— $oc(q, \mathcal{L})=\frac{(6+54)+(6+54)}{2}=60$

On constate donc que $oc(t, \mathcal{L}) < oc(r, \mathcal{L})$ et $oc(q, \mathcal{L}) > oc(r, \mathcal{L})$.

— la stratégie de minimisation que nous utilisons est un nouveau challenge. Nous pourrions penser que minimiser des coûts équivaut à maximiser l'opposé ou l'inverse de ces coûts, et que donc il est possible de ramener cette stratégie aux stratégies existantes. Alors que la première affirmation est vraie, la seconde ne l'est pas. En effet, il n'est pas possible d'utiliser les techniques existantes dans le sens où celles-ci emploient des techniques d'élagage de l'espace de recherche basées sur des bornes supérieures, telles que la Transaction-Weighted Utility (TWU) ou la Sequence Estimated Utility (SEU). La SEU d'une règle r est définie comme la somme des utilités des items pour chaque séquence contenant r [Zida et al., 2015]. Par exemple, en considérant l'event log $\mathcal{L}$ présenté dans le Tableau 3.7, la table de coûts présentée dans le Tableau 3.8 et la règle $r : \{a\} \rightarrow \{b, c\}$, alors $SEU(r)=(1+3+2+3+1+2)+(1+3+2+3+1)=22$. Le coût de r, ou de n'importe quelle règle t avec $r \subset t$, ne pourra donc jamais être plus grand que 22, le SEU étant une borne supérieure. Dans notre cas, il faudrait plutôt définir une borne inférieure pour les différentes traces, ce que nous allons essayer de faire.

On peut donc constater que le problème d'extraction de règles à bas coût comporte plusieurs challenges intéressants. Avec l'algorithme TWINCLE, nous tentons d'y répondre avec la définition de nouvelles techniques d'optimisation.

3.3.3.6 L'algorithme TWINCLE

Dans cette partie, nous introduisons l'algorithme TWINCLE. Nous l'avons développé dans le même esprit que HUSRM [Zida et al., 2015] dans la mesure où nous cherchons à optimiser les temps de calculs par la réduction du nombre de parcours de l'event log et en utilisant une approche "pattern-growth". Dans un premier temps nous détaillons les différentes caractéristiques et optimisations de TWINCLE et présentons l'algorithme de principe dans un second temps.

Contrairement aux algorithmes traditionnels qui comportent une étape dé génération de candidats, nous utilisons l'*expansion* comme méthode de génération de règles. Nous définissons cette méthode comme suit.

Définition 27 - Expansion :
Soit $r : X \rightarrow Y$ *une règle et* σ *une trace. Étendre* r *par une expansion est la tâche qui consiste à ajouter à* r *un item* i *à la fin de l'antécédent ou à la fin du conséquent, afin de générer une nouvelle règle. Un item* i *peut étendre* r *par une expansion à gauche dans* σ $(t : X \cup \{i\} \rightarrow Y)$ *si et seulement si* $\sigma(j) \succ \sigma(k)$, *pour* $\pi_{activity}(\sigma(j))=i, \pi_{activity}(\sigma(k))=x, \forall x \in X, i \notin Y$ *et que* $\sigma(l) \succ \sigma(j)$, *pour* $\pi_{activity}(\sigma(j))=i, \pi_{activity}(\sigma(l))=y, \forall y \in Y$.
Un item i *peut étendre* r *par une expansion à droite dans* σ $(t : X \rightarrow Y \cup \{i\})$ *si et seulement si* $\sigma(j) \succ \sigma(k)$, *pour* $\pi_{activity}(\sigma(j))=i, \pi_{activity}(\sigma(k))=y, \forall y \in Y, i \notin X$.
Notons $droite(r, \sigma)$ *l'ensemble d'items qui peuvent étendre la règle* r *dans* σ *par une expansion à droite et* $gauche(r, \sigma)$ *l'ensemble d'items qui peuvent étendre la règle* r *dans* σ *par une expansion à gauche.*

Soit σ la première trace de l'event log présenté dans le Tableau 3.7, la règle r : $\{a\}\rightarrow\{c,g\}$ peut être étendue à gauche pour créer la règle t : $\{a,b\}\rightarrow\{c,g\}$ ou à droite pour créer la règle q : $\{a\}\rightarrow\{c,g,e\}$. L'item f quand à lui ne peut pas être utilisé pour étendre r à gauche, étant donné qu'il apparaît après tous les items de l'antécédent mais pas avant tous les items de Y ; ni pour étendre r à droite étant donné qu'il n'apparaît pas après tous les items de Y.

Nous introduisons une nouvelle structure, la *table-coût*, inspirée de l'*utility-table* définie dans [Zida et al., 2015], pour calculer efficacement le coût global et le support des différentes règles. Nous définissons la table-coût comme suit.

Définition 28 - Table-Coût :
La table-coût d'une règle r dans un event log $\mathcal{L}$, $ct(r,\mathcal{L})$ est définie comme l'ensemble des tuples tel qu'il existe un tuple (tid, oc, end(X), beg(Y), end(Y)) pour chaque trace contenant r, $\forall\sigma_i \in tra(r)$. tid représente l'indice de la trace dans $\mathcal{L}$, oc représente le coût global de r dans σ_i.
Pour une règle r et une trace σ, end(X) est la position dans σ de l'event dont l'activité est celle représentée par le dernier item de X, end(X)=i où $\pi_{activity}\sigma(i) = x_{|X|}$. beg(Y) est la position dans σ de l'event dont l'activité est celle représentée par le premier item de Y, beg(Y)=i où $\pi_{activity}\sigma(i) = y_1$. end(Y) est la position dans σ de l'event dont l'activité est celle représentée par le dernier item de Y, end(Y)=i où $\pi_{activity}\sigma(i) = y_{|Y|}$. end(X), beg(Y) et end(Y) sont des indices qui vont servir à optimiser le parcours de l'event log lors des différentes expansions.

La table-coût permet d'évaluer rapidement le support absolu d'une règle, qui est simplement égal au nombre de tuples dans la table-coût. Pour les règles de taille 1*1, un premier parcours de l'event log est nécessaire pour construire les différentes tables-coûts. Mais l'efficacité des tables-coût réside dans le fait que pour générer et évaluer les expansions d'une règle r, il suffit de seulement parcourir les traces dont l'indice est stocké dans chaque tuple de la table-coût de r.

Considérons l'event log $\mathcal{L}$ présenté dans le Tableau 3.7 et k=2. La table-coût de la règle r : $\{a\}\rightarrow\{b\}$, $ct(r,\mathcal{L})$ est présentée dans le Tableau 3.9a. Elle est composée de 3 tuples, un pour chaque trace contenant r ; i.e., σ_1, σ_2 etσ_4. Le coût global de la règle se calcule en faisant la moyenne des $oc(r,\sigma)$ de chaque tuple, soit $oc(r,\mathcal{L})=\frac{8+20+8}{3}=12$.
A partir de là, pour effectuer les expansions de r il suffit simplement de parcourir chaque trace dans la table-coût pour sélectionner les activités qui vont étendre r. Par exemple, pour une expansion à gauche de r à partir de la trace σ_2, il suffit d'explorer les items de $gauche(r,\sigma_2)$, soit les items représentant les events de σ_2, du 2^e event (end(X)+1) au 3^e event (beg(Y)-1), soit {d,c}. Similairement, pour une expansion à droite de r à partir de la trace σ_1, il suffit d'explorer les items de $droite(r,\sigma_1)$, soit les items représentant les events de σ_1, du 3^e event (end(Y)+1) au dernier event de σ_1, soit {c,f,g,e}.

Un exemple d'expansion de r à droite est la règle t:$\{a\}\rightarrow\{b,f\}$, dont la table-coût est présentée dans le Tableau 3.9b. Pour calculer rapidement $oc(t,\sigma_i)$ sans avoir à tout réévaluer, il suffit de prendre $oc(r,\sigma_i)$, y ajouter le coût de l'event dont l'activité est représentée par l'item ajouté, ainsi que k fois la différence entre le timestamps de l'event dont l'activité est représentée par le dernier item du conséquent de r et le timestamp de l'event dont l'activité est représentée par l'item ajouté, soit pour t:$X'\rightarrow Y'$: $oc(t,\sigma_i)$=$oc(r,\sigma_i)$ + $\pi_{cost}(\sigma(end(Y'))) + k\cdot(\pi_{timestamp}(\sigma(end(Y'))) - \pi_{timestamp}(\sigma(end(Y')-1)))$. Par exemple, $oc(t,\sigma_1)$=$8+3+2\cdot(8-3)$=21.

tid	oc(r)	end(X)	beg(Y)	end(Y)
σ_1	8	1	2	2
σ_2	20	1	4	4
σ_4	8	1	2	2

(a)

tid	oc(r)	end(X)	beg(Y)	end(Y)
σ_1	21	1	2	4
σ_4	15	1	2	4

(b)

TABLEAU 3.9 – Table-coût des règles *(a)* $\{a\}\rightarrow\{b\}$ et *(b)* $\{a\}\rightarrow\{b, f\}$ dans l'event log de la Figure 3.7.

L'utilisation des tables-coûts permet donc qu'une grande partie de l'event log ne soit pas parcourue, ce qui réduit considérablement les temps de calculs. Nous avons vu à travers l'Expansion et l'utilisation des tables-coûts comment générer de nouvelles règles et évaluer rapidement leur support et coût global. Afin de rendre l'algorithme efficient, il faut penser à réduire l'espace de recherche. Pour cela, nous définissons trois méthodes d'élagage.

Propriété 1 - Élagage par la confiance :
Considérons un log $\mathcal{L}$, un ensemble fini d'activités A et une règle $r : X\rightarrow Y$. Nous pouvons affirmer que si la confiance de r est inférieure au seuil de confiance prédéfinie, $conf_{\mathcal{L}}(r, \mathcal{L}) < minconf$, alors toute règle t générée à partir d'une expansion à droite de r, $t : X\rightarrow Y \cup \{a\}, \forall a \in A$, aura une confiance inférieure au seuil de confiance prédéfinie, $conf_{\mathcal{L}}(t, \mathcal{L}) < minconf$.

Preuve :
Pour un log $\mathcal{L}$ et un ensemble fini d'activités A, le support d'un itemset I est supérieur ou égal au support de chacun de ses sur-ensembles, $sup_{\mathcal{L}}(I) \geq sup_{\mathcal{L}}(J), \forall I \subseteq J$. Considérons une règle $r : X\rightarrow Y$ et une règle t générée à partir d'une expansion à droite de r, $t : X\rightarrow Y\cup \{a\}, \forall a\in A$. L'itemset de r est donc un sous-ensemble de l'itemset de t, $\Omega(r) \subseteq \Omega(t)$. Le support de r est donc supérieur ou égal au support de t, $sup_{\mathcal{L}}(\Omega(r)) \geq sup_{\mathcal{L}}(\Omega(t))$. Comme chacune de ces valeurs est divisée par la même valeur ; i.e., le support de X, pour obtenir la confiance de r et t, nous pouvons affirmer que la confiance de r est supérieure ou égale à la confiance de t, $conf_{\mathcal{L}}(r) \geq conf_{\mathcal{L}}(t)$.

Propriété 2 - Élagage par le coût de la règle :
Considérons une règle r, un event log $\mathcal{L}$ et un ensemble fini d'activités A. Si le coût de r est supérieur au seuil de coût maximum prédéfini, $c(r, \mathcal{L}) > oc_{max}$, alors r et toute expansion à gauche et à droite de r a un coût global supérieur au seuil de coût maximum prédéfini, $oc(r, \mathcal{L}) > oc_{max}$; $oc(t, \mathcal{L}) > oc_{max}$, $t\in \bigcup_{a\in A}(X\cup\{a\}\rightarrow Y) \cup (X\rightarrow Y\cup\{a\})$.

Preuve :
Le coût global d'une règle (OC) est la somme des coûts de chaque items qui la composent (C), auxquels est ajoutée la durée de la règle (T). Si le coût de la règle dépasse le seuil de coût maximum prédéfini, alors le coût global de la règle dépassera nécessairement ce seuil. De plus, le coût d'une règle r étant inférieur ou égal à celui de n'importe laquelle de ses expansions t, si le coût de r dépasse le seuil de coût maximum prédéfini, le coût de t (et donc par extension le coût global de t) le dépassera aussi.

Propriété 3 - Élagage par le coût minimum :
Considérons une règle r, un event log $\mathcal{L}$ et un ensemble fini d'activité A. Nous définissons $min(tra(r))$ comme la trace σ ayant le coût global le plus bas parmi les traces de l'ensemble

$tra(r)$. Si le coût global de cette trace est supérieur au seuil de coût minimum prédéfini, $oc(r, min(tra(r))>oc_{max}$, alors r et toute expansion à gauche et à droite de r a un coût global supérieur au seuil de coût maximum prédéfini, $oc(r, \mathcal{L}) > oc_{max}$; $oc(t, \mathcal{L}) > oc_{max}$, $t \in \bigcup_{a \in A}(X \cup \{a\} \rightarrow Y) \cup (X \rightarrow Y \cup \{a\})$.

Preuve :
La moyenne d'un ensemble de valeurs est nécessairement supérieure ou égale à la plus petite valeur de l'ensemble. Par conséquent pour une règle r, si le coût global le plus bas parmi l'ensemble des traces contenant r est supérieur au seuil de coût maximum prédéfini, alors le coût global de r sur le log sera nécessairement supérieur à ce seuil.

Nous avons vu que la table-coût présentée dans la Définition 28 permet d'évaluer rapidement le support d'une règle. Cependant, l'évaluation de la confiance d'une règle est plus compliquée. Comme nous l'avons expliqué, nous choisissons dans nos travaux d'utiliser des SSR plutôt que des POSR. Un avantage majeur des POSR est le fait qu'elles rendent possible l'utilisation de *bit vectors* pour l'évaluation des itemsets. Le bit vector $bv(i)$ d'un item i dans un log $\mathcal{L}$ contient $|\mathcal{L}|$ bits, où le j^e bit est égal à 1 si i apparaît dans la trace σ_j, 0 sinon. Par exemple, pour $\mathcal{L} = \langle \sigma_1, \sigma_2, \sigma_3, \sigma_4 \rangle$ le log présenté dans le Tableau 3.7, $bv(b)$=1101 et $bv(f)$=1011. Pour calculer le support d'un itemset I, il suffit de faire l'intersection des bit vectors de chaque item i dans I, $bv(I) = \bigcap_{i \in I} bv(i)$. Par exemple, $bv(bf)$=1101∩1011 = 1001. Cette méthode, utile pour évaluer la confiance d'une règle, est très efficace car l'intersection de bit vectors est une opération rapide. De plus, les bit vectors sont des structures qui ne consomment pas beaucoup de ressources mémoire.

Malheureusement, dans le cas des SSR, l'utilisation des bit vectors n'est pas possible. Le bit vector d'un itemset I indique si les différents items de I apparaissent dans les différentes traces, mais ne donne aucune indication sur l'ordre dans lequel ils apparaissent. Une méthode naïve pour l'évaluation des différents itemsets serait de parcourir le log pour chaque règle. Pour pallier cette limite, nous proposons de mémoriser les supports déjà calculés. De nos jours, la consommation de ressources mémoire est de moins en moins un problème, étant donné que les ordinateurs les plus standards sont souvent équipés de plusieurs Gigabits de mémoire RAM. L'enjeu est de stocker efficacement des supports déjà calculés pour une courte période, et ainsi réduire considérablement les temps de calculs lors de l'évaluation de la confiance des différentes règles.

Définition 29 - Évaluation du support par l'itemset de la règle :
Au cours de l'exécution de l'algorithme, le support de certaines règles est stocké dans une HashMap, une structure de données qui permet un accès rapide. Ce choix est motivé par le fait que le support d'une règle $r : X \rightarrow Y$ peut servir à évaluer la confiance d'une règle $t : X' \rightarrow Y'$ avec $X \cup Y = X'$. De cette façon, stocker intelligemment certains supports dans la HashMap permettra un accès rapide pour récupérer le support de l'antécédent d'une règle.

Jusque là, nous avons défini les différentes structures et méthodes développées pour optimiser les temps de calcul lors de la fouille. Parmi ces optimisations, celles présentées dans les Définitions 27 et 29, et dans la Propriété 1 nécessitent que l'algorithme soit structuré d'une certaine manière pour que leur mise en place soit possible. Nous détaillons cet aspect dans la suite de cette sous-section.

L'*Expansion* est définie comme le principe d'étendre une règle r en ajoutant un item a l'extrémité de l'antécédent pour une expansion à gauche, ou à l'extrémité du conséquent pour une expansion à droite, pour générer une nouvelle règle t. Nous présentons ce principe

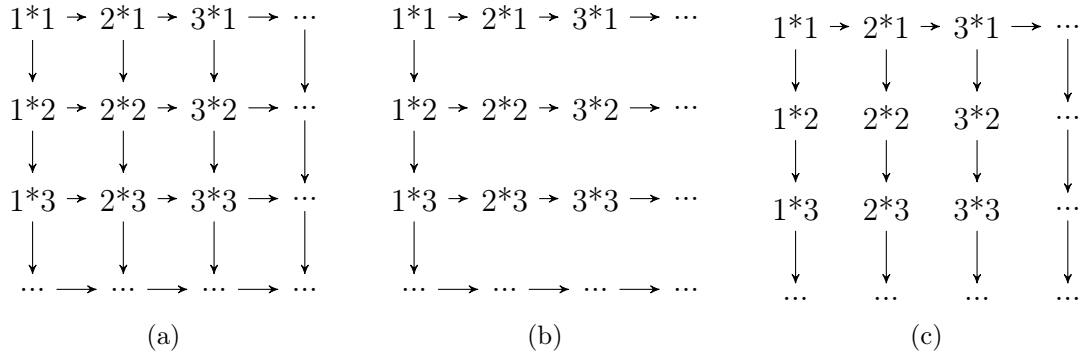


FIGURE 3.20 – Principes *(a)* d'expansion totale *(b)* d'expansion limitée à gauche et *(c)* d'expansion limitée à droite.

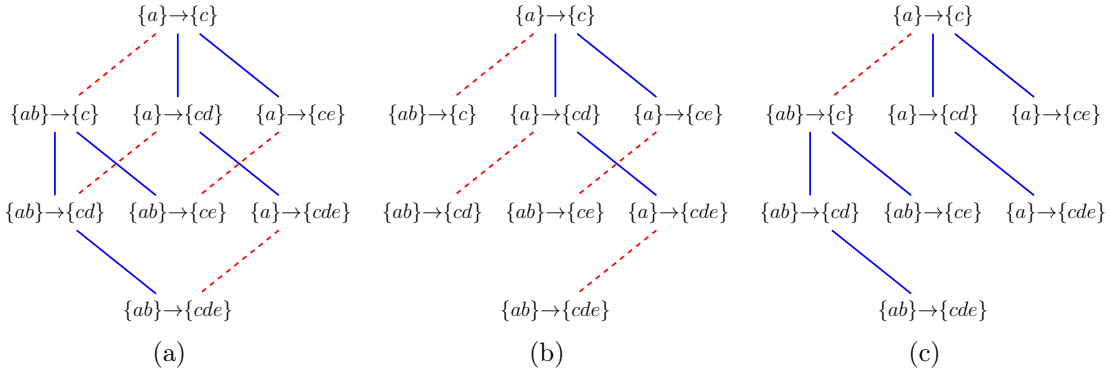


FIGURE 3.21 – *(a)* Exemples d'expansion totale, *(b)* d'expansion limitée à gauche et *(c)* d'expansion limitée à droite à partir de la règle $\{a\}\rightarrow\{c\}$ et la séquence $\langle a, b, c, d, e\rangle$.

d'expansion dans la Figure 3.20. Tout en haut à gauche, les règles de taille 1*1 sont générées. Une expansion à gauche (représentée par les flèches allant vers la droite), ajoute un item à l'antécédent de ces règles pour former des règles de tailles 2*1. Parallèlement, une expansion à droite (représentée par les flèches allant vers le bas) ajoute un item au conséquent de la règle pour former des règles de taille 1*2. Cette procédure d'expansion est répétée tant qu'il est possible de générer de nouvelles règles. Nous constatons que, si cette méthode d'expansion totale est appliquée, il est possible de générer une règle r de taille $k * m$ à partir d'une règle t de taille $(k-1) * (m-1)$ en appliquant successivement une expansion à gauche puis une expansion à droite, ou alternativement en appliquant d'abord une expansion à droite puis une expansion à gauche. Tel quel, l'algorithme peut donc générer des règles en doublons. Pour pallier ce problème, une technique est de limiter l'expansion à gauche ou bien à droite. Limiter l'expansion à gauche signifie qu'il est interdit d'effectuer une expansion à droite après une expansion à gauche (Figure 3.20b) et limiter l'expansion à droite signifie qu'il est interdit d'effectuer une expansion à gauche après une expansion à droite (Figure 3.20c). De cette façon, il est impossible de générer deux fois la même règle.

Les principes d'expansion totale, d'expansion limitée à gauche et d'expansion limitée à droite sont illustrés respectivement dans les Figures 3.21a, 3.21b et 3.21c en considérant $r : \{a\}\rightarrow\{c\}$ comme règle de départ et la séquence $\langle a, b, c, d, e\rangle$ pour définir les différents items disponibles pour les expansions. Une expansion à gauche est représentée par un trait en pointillé et une expansion à droite est représentée par un trait plein.

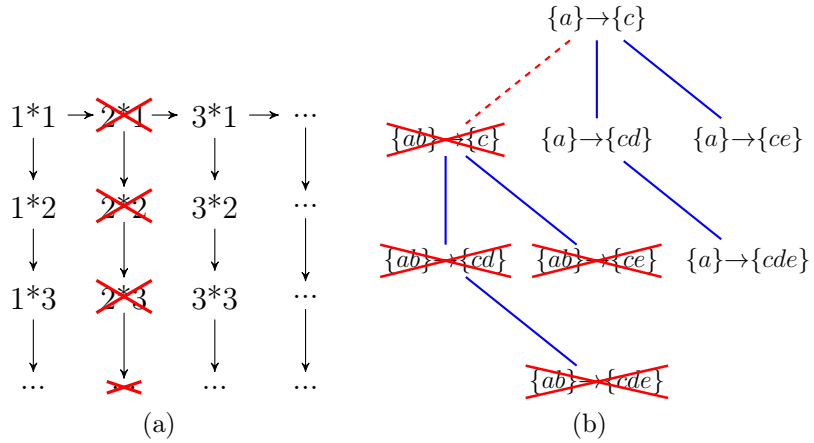


FIGURE 3.22 – *(a)* Principe de l'élagage par expansion à droite et *(b)* exemple à partir de la Figure 3.21c.

Si le choix entre une limitation à gauche ou à droite peut sembler à première vue arbitraire, limiter l'expansion à droite va nous permettre de rendre l'élagage par la confiance (Propriété 1) beaucoup plus efficace que si nous avions opté pour une expansion limitée à gauche. Nous présentons dans la Figure 3.22a le principe de l'élagage à partir de la confiance sur le principe d'expansion à droite. Considérons que la confiance d'une règle $\{ab\}\rightarrow\{c\}$ soit inférieure au seuil minimum de confiance prédéfini. D'après la Propriété 1, il est inutile d'étendre cette règle à droite. Cette propriété nous permet donc de supprimer toute la "branche" d'expansion à droite de cette règle, et ainsi éviter de générer beaucoup de règles inutilement. Cette optimisation ne pourrait s'appliquer dans le cadre d'une expansion limitée à gauche car il n'y a pas de lien entre une règle de taille k*m et ses extensions de taille k*(m+1), excepté si la règle en question est de taille 1*m, auquel cas il est possible de supprimer la première "branche" d'expansion à droite.

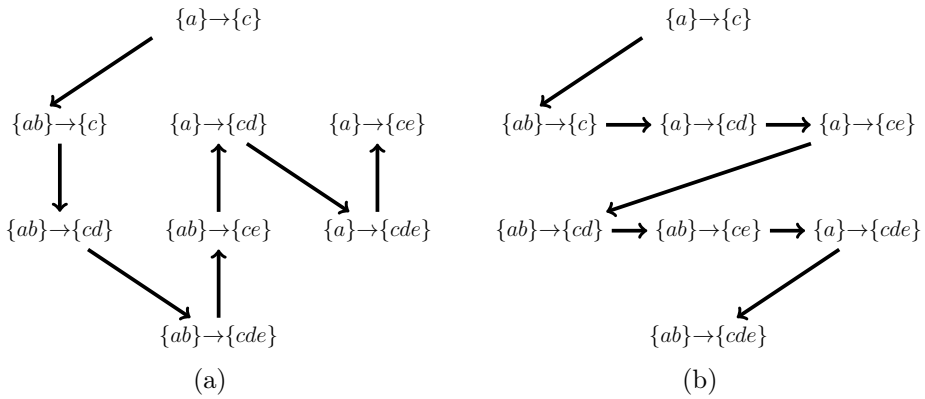


FIGURE 3.23 – *(a)* Déroulement du parcours Depth First Search (DFS) et *(b)* du parcours Breadth First Search (BFS) sur l'arbre d'expansion limitée à droite de la Figure 3.21c.

La Définition 29 présente une méthode d'évaluation de l'antécédent d'une règle basée sur l'accès aux supports de règles précédemment calculés stockés dans une HashMap. Cette évaluation nécessite donc que pour toute règle $r : X\rightarrow Y$ dont nous voulons calculer la confiance, le support de la règle $t : X'\rightarrow Y'$ avec $X=X'\cup Y'$ ait déjà été calculé. Dans les algorithmes tels que TRuleGrowth ou HUSRM [Fournier-Viger et al., 2015, Zida et al.,

2015], l'expansion des règles est faite de façon récursive avec un parcours en profondeur (*Depth First Search* ou DFS). Dans l'exemple de la Figure 3.21c, la règle $\{a\}\rightarrow\{c\}$ est d'abord générée. Puis la règle $\{ab\}\rightarrow\{c\}$, $\{ab\}\rightarrow\{cd\}$ et $\{ab\}\rightarrow\{cde\}$. S'il n'est plus possible de générer de nouvelles règles à partir d'une règle, il faut remonter jusqu'à pouvoir générer de nouvelles règles. Nous présentons le principe de DFS sur cet exemple dans la Figure 3.23a. Cependant, ce parcours ne garantit pas que le support d'une règle dont l'itemset est l'antécédent d'une règle dont nous voulons évaluer la confiance ait déjà été calculé. Pour cela, nous procédons avec un parcours en largeur (*Breadth First Search* ou BFS). Avec ce parcours, utilisé par les techniques traditionnelles avec génération de candidats telles qu'Apriori [Agrawal et al., 1993], chaque groupe de règles de taille $k*m$ avec $k+m{=}p$ est évalué. Puis une série d'expansions est effectuée sur ces règles pour générer les règles de taille $k*m$ avec $k*m{=}p+1$ et ainsi de suite. De cette façon, nous sommes sûrs d'avoir déjà calculé le support d'une règle dont l'itemset représente l'antécédent d'une règle dont nous voulons évaluer la confiance. Dans l'exemple de la Figure 3.21c, la règle $\{a\}\rightarrow\{c\}$ est d'abord générée. Puis les règles $\{ab\}\rightarrow\{c\}$, $\{a\}\rightarrow\{cd\}$ et $\{a\}\rightarrow\{ce\}$. Quand il n'est plus possible de générer de règles de taille p, alors nous pouvons générer les règles de taille $p+1$. Nous présentons le principe de BFS sur cet exemple dans la Figure 3.23b.

Nous présentons le pseudo code de TWINCLE dans l'Algorithme 2. D'abord, il parcourt le log pour construire R l'ensemble de toutes les règles de taille 1*1 respectant la fenêtre de temps. Ensuite, TWINCLE effectue de façon itérative l'évaluation et l'expansion des règles comme suit. Pour chaque règle r de taille $k*m$ avec $k+m{=}p$ (en commençant donc avec $k{=}m{=}1$), si le coût et le coût minimum sont inférieurs au seuil de coût maximum prédéfini (Propriétés 2 et 3), alors nous regardons si les différentes mesures de coût global, de cohérence, de confiance et de longueur de r respectent leur seuil respectif prédéfini. Pour des raisons de lisibilité, nous avons défini la fonction $isLong(r) = (|X| \geq \mu X) \wedge (|Y| \geq \mu Y) \wedge (|\Omega(r)| \geq \mu\Omega)$. Si r respecte chacun de ces seuils, alors c'est une règle séquentielle à bas coût avec respect des contraintes, nous pouvons donc la sauvegarder ou l'afficher. Ensuite, dans tous les cas, la procédure *expansionGauche*() est appelée pour générer toutes les règles avec une expansion à gauche à partir de r. Par contre, la procédure *expansionDroite*() est appelée seulement si la confiance de r est supérieure ou égale au seuil de confiance minimum prédéfini ; sinon l'expansion à droite de la règle (Propriété 1) est arrêtée. Cette partie est effectuée pour chacune des règles de taille 1*1 ou générée à partir d'une expansion à gauche. Pour les règles générées à partir d'une expansion à droite, seule la partie relative au test de la Propriété 1 pour l'appel de la procédure *expansionDroite*() est conservée, comme une expansion à gauche est interdite après une expansion à droite. Enfin toutes les règles de taille $n*q$ avec $n+q{=}p+1$ qui respectent les contraintes sont générées, et l'algorithme recommence la boucle d'évaluation/génération.

Algorithme 2 : Algorithme TWINCLE

Entrées : $\mathcal{L}$: un event log ; minconf, oc_{max}, tw_{max}, coh_{max} : les différents seuils
Parcourir $\mathcal{L}$ pour générer R, l'ensemble de règles r de la forme $\{x\}\rightarrow\{y\}$ $(x, y \in \mathcal{A})|tw(t,\sigma) \leq twmax$
$\mathrm{R}_L := \mathrm{R}$;
$\mathrm{R}_R := \emptyset$;
tant que $R_R \neq \emptyset$ *ou* $R_L \neq \emptyset$ **faire**
  $\mathrm{R}^*_R := \mathrm{R}^*_L := \emptyset$;
  pour tous $r \in R_L$ **faire**
    si $c(r,\mathcal{L}) \leq oc_{max}$ *(Propriété 2) et* $oc(r, min(tra(r))) \leq oc_{max}$ *(Propriété 3)* **alors**
      si $conf(r) \geq minconf$ *(Propriété 1)* **alors**
        si *isLong(r) et* $oc(r,\mathcal{L}) \leq oc_{max}$ *et* $coh(r,\mathcal{L}) \leq coh_{max}$ **alors**
          afficher r
        $R^*_R = R^*_R \cup expansionDroite(r, ct(r,\mathcal{L}))$;
      $R^*_L = R^*_L \cup expansionGauche(r, ct(r,\mathcal{L}))$;
  pour tous $r \in R_R$ **faire**
    si $c(r,\mathcal{L}) \leq oc_{max}$ *(Propriété 2) et* $oc(r, min(tra(r))) \leq oc_{max}$ *(Propriété 3)* **alors**
      si $conf(r) \geq minconf$ *(Propriété 1)* **alors**
        si *isLong(r) et* $oc(r,\mathcal{L}) \leq oc_{max}$ *et* $coh(r,\mathcal{L}) \leq coh_{max}$ **alors**
          afficher r
        $R^*_R = R^*_R \cup expansionDroite(r, ct(r,\mathcal{L}))$;
  $\mathrm{R}_L := \mathrm{R}^*_L$;
  $\mathrm{R}_R := \mathrm{R}^*_R$;

Algorithme 3 : Algorithme expansionGauche

Entrées : r : une règle ; $ct(r, \mathcal{L})$: la table de coûts de r
R := $\emptyset$
pour tous $\sigma \in tra(r)$ *appartenant à* $ct(r, \mathcal{L})$ **faire**
    pour tous $t : X \cup \{a\} \rightarrow Y | a \in gauche(r, \sigma)$ **faire**
        $R \leftarrow R \cup \{t\}$;
        Mettre à jour ct(t) ;
Retourner R ;

Algorithme 4 : Algorithme expansionDroite

Entrées : r : une règle ; $ct(r, \mathcal{L})$: la table de coûts de r
R := $\emptyset$
pour tous $\sigma \in tra(r)$ *appartenant à* $ct(r, \mathcal{L})$ **faire**
    pour tous $t : X \rightarrow Y \cup \{a\}, a \in droite(r, \sigma) | tw(t, \sigma) \leq twmax$ **faire**
        $R \leftarrow R \cup \{t\}$;
        Mettre à jour ct(t) ;
Retourner R ;

Conclusion

En prenant en compte les enjeux liés à la modélisation et aux différents risques que pose l'utilisation de données issues de systèmes d'information, nous avons proposé dans ce chapitre *KITE* une méthodologie innovante qui repose sur le pouvoir d'expression des modèles de processus pour effectuer une extraction de connaissance plus intéressante.

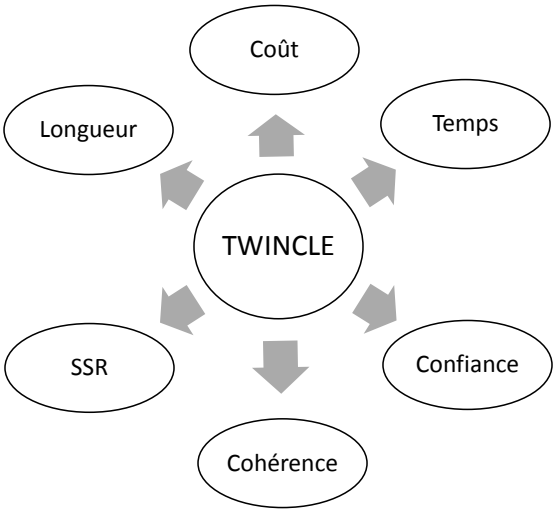


FIGURE 3.24 – Caractéristiques de l'algorithme TWINCLE.

Afin de répondre aux problématiques posées dans le chapitre introductif de ce manuscrit, nous avons proposé deux instanciations.

Dans ce manuscrit, nous avons présenté *PRISM*, dont l'objectif est d'extraire des dépendances pertinentes entre activités. Après avoir construit un modèle avec l'Inductive Miner, un cycle de sélection/réduction est effectué pour réduire l'espace de recherche à explorer. Dans cette instanciation, nous représentons les relations de dépendances sous forme de règles séquentielles à bas coût. Par conséquent, dans la troisième étape de KITE, nous avons proposé l'algorithme *TWINCLE*. Nous présentons ces principales caractéristiques dans la Figure 3.24. Le coût est un nouvelle mesure que nous avons introduite et qui représente la "pénalité" impliquée par chaque item. Dans nos travaux nous cherchons à minimiser cette mesure. Le temps est une contrainte que nous prenons en compte pour interpréter au mieux les dépendances extraites ; et la longueur est une contrainte utilisée notamment pour réduire le nombre de dépendances extraites. La confiance est une mesure classique utilisée pour évaluer la robustesse des dépendances. La cohérence est une mesure que nous avons proposée pour évaluer dans quelle mesure la dépendance était "maîtrisée" ; i.e., avait un coût et une durée sensiblement identique dans chacune des traces dans lesquelles elle apparaît. Enfin, nous utilisons les règles séquentielles standards (SSR) contrairement aux règles séquentielles partiellement ordonnées (POSR) car nous considérons que l'ordre des items dans l'antécédent et le conséquent des règles est important en termes de significations médicales.

La seconde instanciation que nous avons développée est *IDM*. Cette instanciation est une version relâchée de PRISM, dans laquelle est seulement analysé l'impact des choix sur le processus, et non plus l'impact de l'ordre de ces choix. Les principales différences avec PRISM sont les suivantes :

— seuls les choix exclusifs sont pris en compte

— les dépendances ne concernent que deux activités

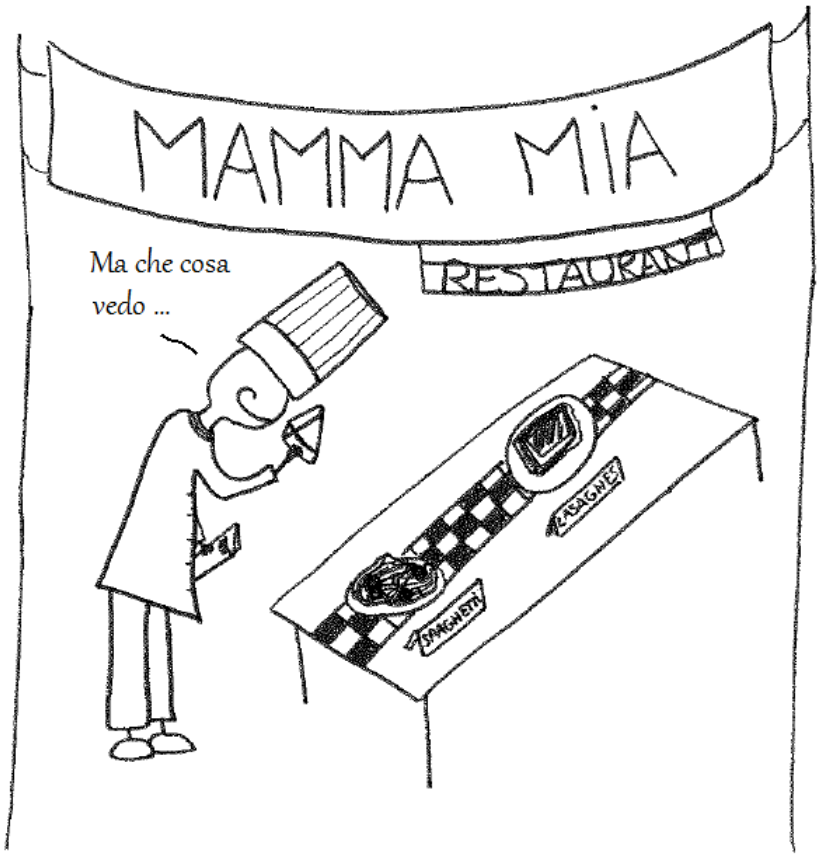
Dans IDM, nous construisons le modèle dans l'étape #1 avec l'*Evolutionary Tree Miner* [Buijs, 2014], un algorithme génétique qui construit un modèle sous la forme d'un process tree. Dans l'étape #2, nous avons proposé un parcours intelligent du process tree pour détecter les activités entre lesquelles il pouvait y avoir une dépendance. Enfin, dans l'étape #3, nous vérifions l'existence de dépendances via l'extraction de règles d'association. Lorsqu'il s'agit d'analyser seulement l'impact des choix entre deux activités dans le processus, IDM est un choix plus adapté car le périmètre restreint sur lequel il intervient lui permet d'être plus rapide.

Dans le chapitre suivant, nous effectuons l'étape #4 de la méthodologie, relative à l'interprétation des résultats, en appliquant les différentes méthodes proposées dans ce chapitre et en évaluant leurs performances.

Chapitre 4

Mise en œuvre des propositions

Sommaire



Introduction

Dans le chapitre précédent, nous avons introduit la méthodologie *KITE* ainsi que *PRISM*, une instanciation de la méthodologie pour l'extraction de dépendances profitables entre activités. PRISM intègre *TWINCLE*, un algorithme que nous avons développé pour l'extraction de règles séquentielles à bas coût. Ce chapitre a pour objectif d'appliquer la quatrième étape de la méthodologie KITE relative à l'interprétation des résultats.

Ce chapitre est organisé comme suit. Dans la section 4.1, nous effectuons des expérimentations sur des jeux de données issus de la littérature pour mesurer les performances de l'algorithme TWINCLE.

Dans la section 4.2, nous effectuons deux études de cas pour évaluer les performances de l'instanciation PRISM en termes de temps d'exécution et de qualité des dépendances extraites. La première étude de cas porte sur un log généré dans l'optique de représenter un processus structuré. La seconde étude de cas porte sur un log construit à partir de données extraites des systèmes d'information du Centre Hospitalier du Pays de Craponne-sur-Arzon.

4.1 Analyse de performance de l'algorithme TWINCLE

Pour mesurer les performances de TWINCLE, nous avons effectué des expérimentations sur 4 logs que nous avons générés à partir de datasets connus de la littérature relative à l'extraction de motifs.

Log	Nombre d'items	Nombre de traces	Nb moy(items/trace)
BIBLE	13 905	36 369	17,84
SIGN	267	730	51,99
KOSARAK	10 094	10 000	8,14
FIFA	2 990	20 450	34,74

TABLEAU 4.1 – Caractéristiques des logs issus de la littérature utilisés pour les expérimentations.

Nous présentons les caractéristiques principales de ces datasets dans le Tableau 4.1. Le dataset *FIFA* contient des données de navigation sur le site web de la FIFA lors de la coupe du monde 1998. Il est composé de 2 990 items distincts et de 20 450 séquences d'une longueur moyenne de 34,75 items. *KOSARAK* est un dataset qui contient des données de navigation sur un portail web d'informations hongrois. La version utilisée est une partie du dataset original et est composée de 10 094 items distincts et de 10 000 séquences d'une longueur moyenne de 8.14 items. Le dataset *BIBLE* est une conversion de la Bible en séquences, dans lesquelles chaque mot est représenté par un item. Il est composé de 13 905 items distincts et 36 369 séquences dont la longueur moyenne est de 17.84 items. *SIGN* est un dataset mis à disposition par le *National Center for Sign Language and Gesture Resources* de l'Université de Boston. Il consiste en un ensemble d'énoncés, où chaque énoncé associe un segment de vidéo avec une transcription détaillée. Il est composé de 267 items distincts et de 730 séquences d'une longueur moyenne de 51.99 items.

Les datasets utilisés sont ceux disponibles sur le site SPMF [1].Ils sont donc composés de séquences, elles même composées d'items associés à un profit. Nous les avons transformés

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

	Expérimentation 1	**Expérimentation 2**	**Expérimentation 3**
oc_{max}	[20 ;180]	150	150
$minconf$	0,7	[0,1 ;0,9]	0,7
tw_{max}	200	200	[50 ;250]
coh_{max}	0,1	0,1	0,1
Objectif	évaluer l'impact de oc_{max}	évaluer l'impact de $minconf$	évaluer l'impact de tw_{max}

TABLEAU 4.2 – Valeur des différents paramètres utilisés pour les trois expérimentations.

en event logs de la façon suivante. Une trace correspond à une séquence. L'item et son profit dans le dataset original ont été utilisés comme attribut "activité" et "coût" dans les events. Après avoir aléatoirement fixé le timestamp du premier event d'une trace, chaque event restant a été généré en l'espaçant de son prédécesseur d'un nombre d'heures aléatoire sur l'intervalle [1,50].

Les expérimentations menées ont pour but d'évaluer les performances de TWINCLE en temps d'exécution et en consommation mémoire, en faisant varier différents paramètres. Les paramètres retenus sont les mesures dont les seuils sont censés impacter la taille de l'espace de recherche. Nous présentons ces expérimentations dans le Tableau 4.2.

La 1re expérimentation a pour but d'évaluer l'impact du coût global maximum oc_{max} en faisant varier sa valeur sur l'intervalle [20 ;180] ; 20 étant la valeur en dessous de laquelle peu, voire aucune règle, n'est extraite et 180 étant la valeur maximum impactant de façon notable les temps de calcul. Cette expérimentation pourra évaluer l'efficacité des Propriétés 2 et 3 ; i.e., les méthodes d'élagage par le coût de la règle et par le coût global minimum de la règle.

La 2e expérimentation a pour but d'évaluer l'impact de la confiance minimum $minconf$ en faisant varier sa valeur sur l'intervalle [0,1 ;0,9]. Cette expérimentation pourra évaluer l'efficacité de la Propriété 1 relative à l'élagage par la confiance de la règle.

Enfin, la 3e expérimentation a pour but d'évaluer l'impact de la fenêtre de temps maximum tw_{max} en faisant varier sa valeur sur l'intervalle [50 ;250], 50 étant la valeur au dessous de laquelle trop peu de règles sont extraites et 250 la valeur au dessus de laquelle les temps de calcul explosent. Même si la fenêtre de temps n'est pas explicitement utilisée dans une technique d'élagage, elle est implicitement utilisée dans les procédures d'expansion pour élaguer l'espace de recherche. En effet, si une règle r n'est pas comprise dans la fenêtre de temps prédéfinie, alors toute expansion à gauche comme à droite de r ne sera pas non plus comprise dans la fenêtre de temps prédéfinie.

Les autres paramètres du Tableau 4.2 ont été fixés soit à partir de valeurs habituellement utilisées dans la littérature, soit empiriquement de sorte qu'un nombre suffisant de règles soit extraites.

Les expérimentations ont été menées sur une machine de calcul pourvue de processeurs Intel Xeon 2.4 GHz et d'1 To de RAM. Pour des raisons de reproductibilité, nous avons contraint TWINCLE à l'utilisation d'un seul cœur et une consommation maximale de 40 Go de RAM. L'objectif est donc d'évaluer comment réagissent les différentes techniques d'élagage (Propriétés 1, 2 et 3) en fonction des valeurs que peuvent prendre ces seuils. Pour cela, nous définissons trois configurations :

1. la configuration cfg_1 représente une exécution de TWINCLE n'utilisant aucune des techniques d'élagage, explorant ainsi tout l'espace de recherche

2. la configuration cfg_2 est une exécution de TWINCLE qui n'utilise que les méthodes d'élagage relatives au coût global (Propriétés 2 et 3)
3. la configuration cfg_3 est une exécution de TWINCLE qui utilise toutes les méthodes d'élagage (Propriétés 1, 2 et 3)

Pour les 4 logs, les Figures 4.1 et 4.2 présentent les temps d'exécution et les Figures 4.3 et 4.4 présentent la consommation mémoire.

Bien qu'avec des ordres de grandeur différents, nous remarquons que les différents tests effectués impactent de la même manière les performances de TWINCLE sur les 4 logs. Globalement, la configuration utilisant toutes les techniques d'élagage est bien meilleure que les deux autres, prouvant que ces techniques trouvent leur utilité.

Au niveau des temps d'exécution, la variation de oc_{max} n'influe pas sur les performances de cfg_1, étant donné que cette configuration n'élague pas à partir du coût global. Sur des petites valeurs de oc_{max}, cfg_2 et cfg_3 réduisent fortement les temps de calcul pour les 4 logs, allant jusqu'à un facteur 23 pour FIFA. Naturellement, les performances de cfg_2 se rapprochent petit à petit de celles de cfg_1 avec l'augmentation de oc_{max}, voire très rapidement par exemple pour BIBLE. Seul cfg_3 maintient des temps de calcul acceptables peu importe la valeur de oc_{max} sur l'intervalle testé.

Comme cfg_1 et cfg_2 n'élaguent pas à partir de la confiance, la variation de $minconf$ n'influe pas sur leur temps de calcul. A l'inverse, cfg_3 qui utilise la Propriété 1 pour réduire l'espace de recherche à partir de la confiance, voit les temps de calcul diminuer au fur et à mesure que $minconf$ se rapproche de 1, avec une réduction allant d'environ 10% pour BIBLE, jusqu'à environ 36% pour KOSARAK.

Enfin, la valeur de la fenêtre de temps maximale est le seuil qui impacte le plus lourdement les temps de calcul, quels que soient le log ou la configuration. La raison est qu'elle modifie la taille de l'espace de recherche. Le nombre de règles à explorer, et par extension le temps de calcul, croissent de façon exponentielle avec l'augmentation de la fenêtre de temps minimale. Certains calculs n'ont pas abouti, ayant consommé plus que l'espace mémoire alloué.

Quelle que soit l'expérimentation menée, la consommation de mémoire suit globalement les même évolutions que les temps de calcul sur les différents logs. Toutefois, deux ordres de grandeur se dégagent. Pour BIBLE et FIFA, la consommation mémoire est particulièrement élevée. Pour des seuils "intermédiaires" (i.e., $oc_{max} = 120$, $minconf = 0.7$, $tw_{max} = 200$), il est à noter une consommation d'environ 3 Go pour BIBLE et d'environ 5 Go pour FIFA, malgré l'utilisation des méthodes d'élagage (cfg_3). Bien que ces résultats soient sensiblement élevés, il est à noter que cfg_3 reste néanmoins bien plus performant qu'une exploration naïve, avec une réduction de la consommation mémoire jusqu'à un facteur d'environ 6 pour FIFA. La consommation mémoire de TWINCLE sur les deux autres logs est plus raisonnable, avec pour les seuils intermédiaires définis ci-dessus, environ 450 Mo pour KOSARAK et 500 Mo pour SIGN pour cfg_3. De façon générale sur ces deux logs, la configuration utilisant les techniques d'élagage consomme au moins deux fois moins de mémoire par rapport à la configuration naïve.

Pour les 4 logs, les Figures 4.5 et 4.6 présentent le nombre de règles qui ont été générées et les différentes configurations de TWINCLE (colonnes cfg_1, cfg_2 et cfg_3) ; ainsi que le nombre de règles extraites (colonne nbEx).

Nous remarquons que de manière générale, lorsque le seuil de coût global est assez bas, le nombre de règles générées diminue fortement, jusqu'à un facteur 118 pour FIFA. Avec les paramètres que nous avons fixés, nous constatons qu'il arrive qu'aucune règle ne

soit extraite lorsque le seuil de coût global est très bas. A l'inverse, lorsque le seuil de coût global augmente, le nombre de règles extraites atteint plusieurs milliers, ce qui est bien plus que ce qu'un humain peut analyser. Pour des seuils "intermédiaires" (i.e., $oc_{max} = 120$, $minconf = 0.7$, $tw_{max} = 200$), nous constatons que l'utilisation des techniques d'optimisation a permis de réduire le nombre de règles générées par un facteur 5,7, 6,6 et 5 pour les logs BIBLE, SIGN et KOSARAK. Pour FIFA, l'utilisation des techniques d'élagage a permis une fouille complète, la configuration sans technique d'optimisation ayant consommé plus de mémoire que la limite que nous avions fixée.

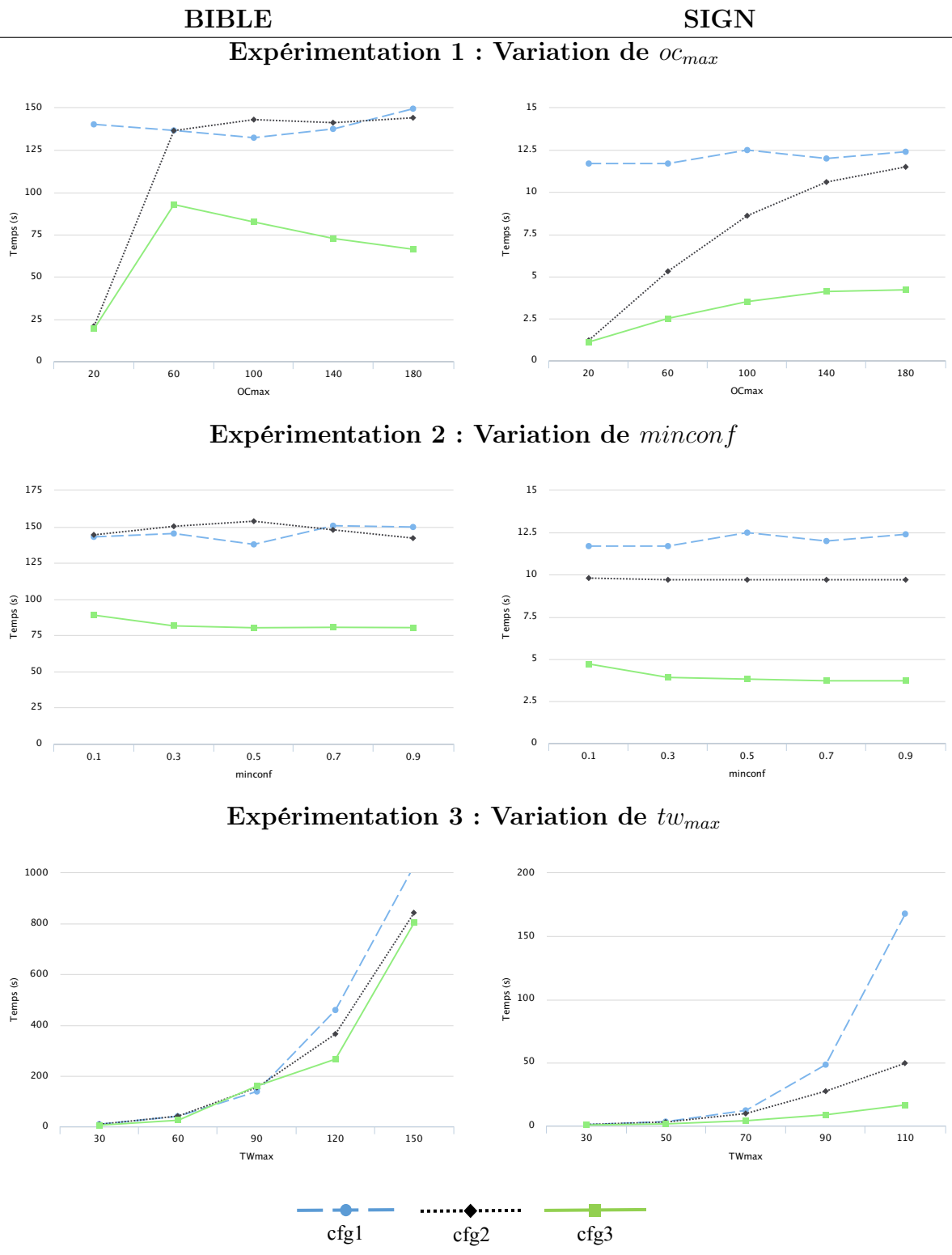


FIGURE 4.1 – Temps d'exécution (en secondes) de TWINCLE sur les logs BIBLE et SIGN en fonction des 3 expérimentations menées.

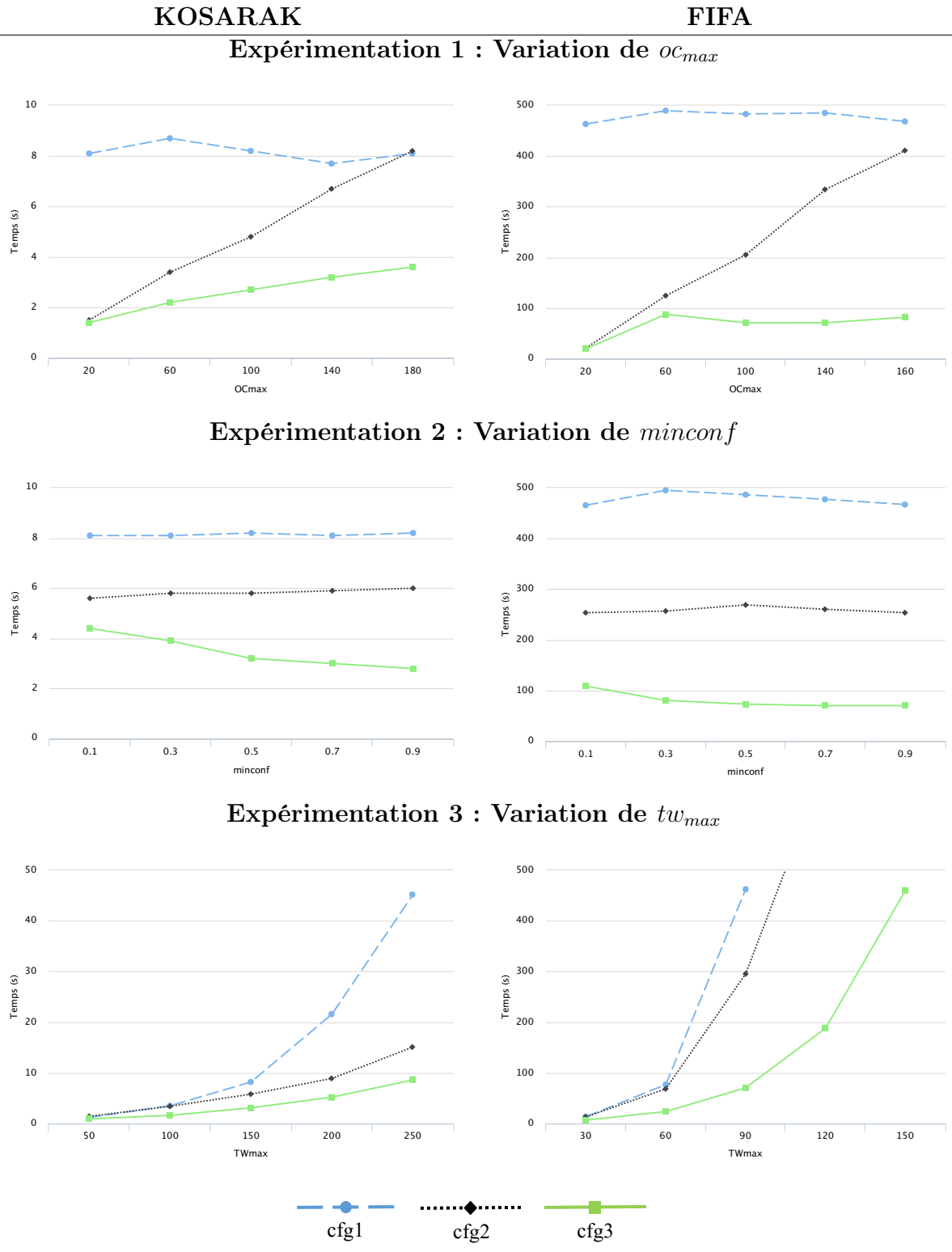


FIGURE 4.2 – Temps d'exécution (en secondes) de TWINCLE sur les logs KOSARAK et FIFA en fonction des 3 expérimentations menées.

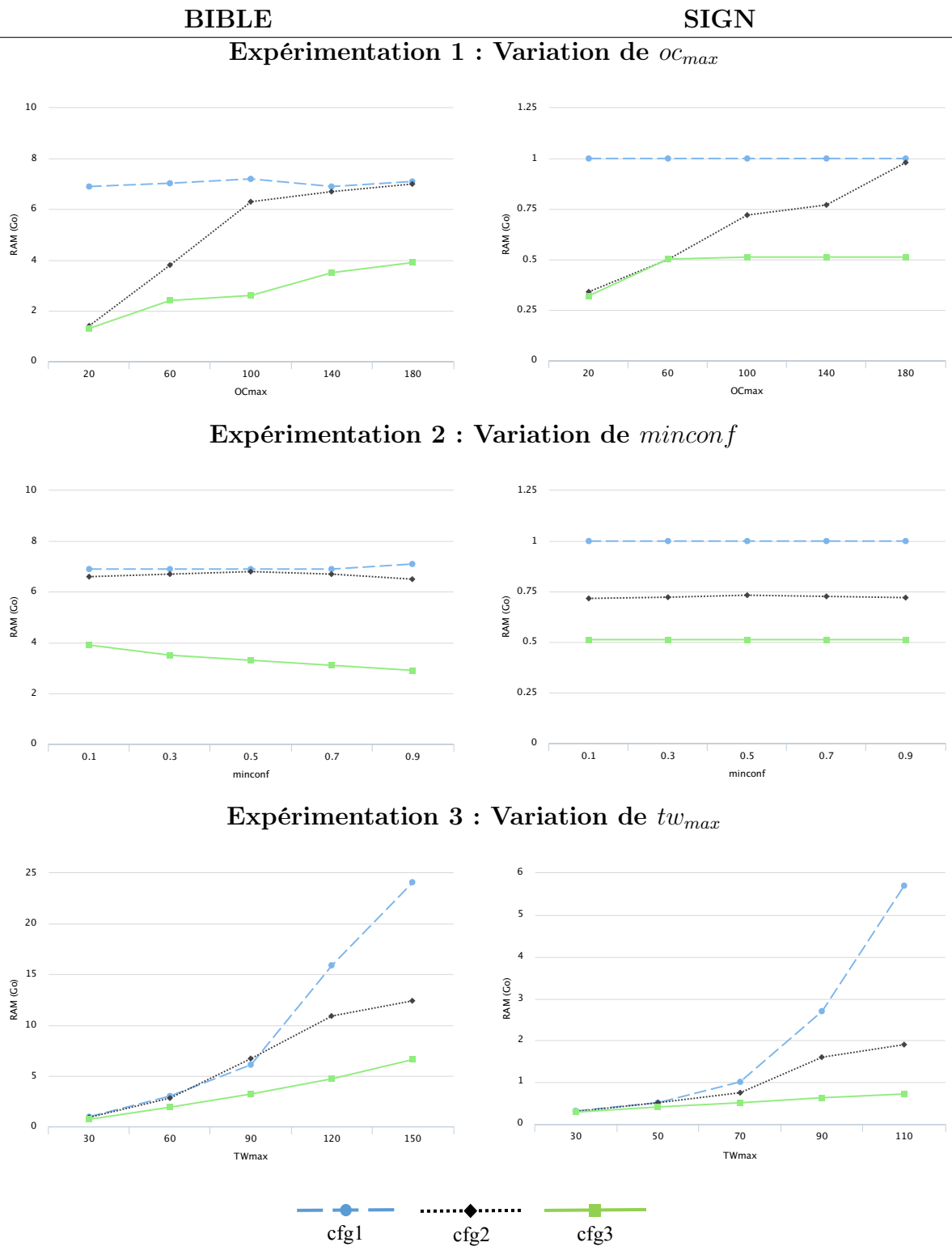


FIGURE 4.3 – Consommation mémoire (en Go) de TWINCLE sur les logs BIBLE et SIGN en fonction des 3 expérimentations menées.

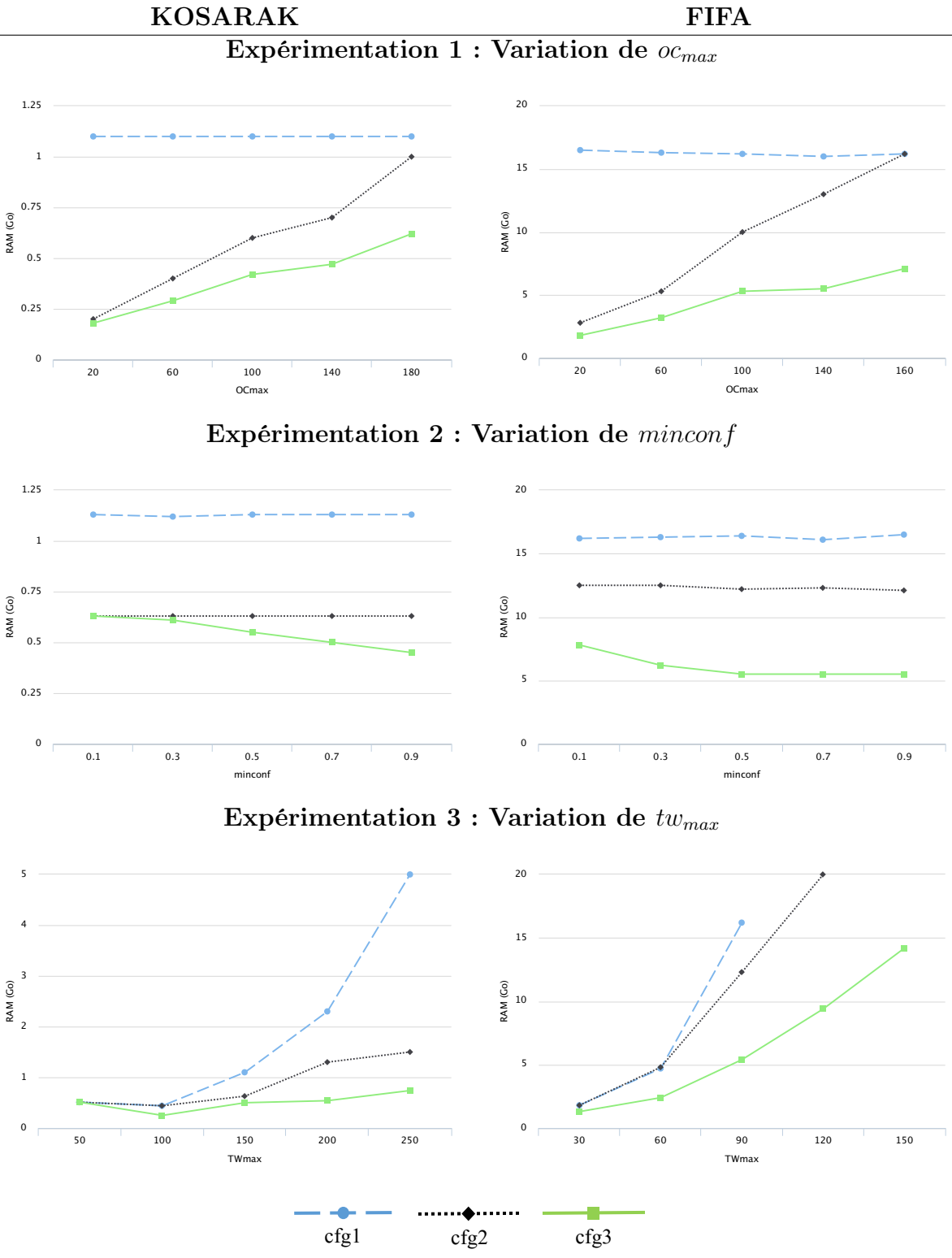


FIGURE 4.4 – Consommation mémoire (en Go) de TWINCLE sur les logs KOSARAK et FIFA en fonction des 3 expérimentations menées.

BIBLE

Test	Valeur	nbEx	cfg_1	cfg_2	cfg_3
Variation oc_{max}	20	2	14 715 953	981 274	772 552
	60	64	14 715 953	4 671 393	2 114 563
	100	1 251	14 715 953	8 637 973	3 322 904
	140	9 520	14 715 953	12 075 365	4 403 776
	180	41 791	14 715 953	14 066 753	5 136 346
Variation $minconf$	0,1	17 262	14 715 853	10 482 257	5 276 615
	0,3	9 567	14 715 953	10 482 257	4 306 121
	0,5	7 168	14 715 953	10 482 257	4 052 062
	0,7	3 677	14 715 953	10 482 257	3 885 772
	0,9	3 530	14 715 953	10 482 257	3 833 131
Variation tw_{max}	50	784	731 353	748 662	448 533
	100	4 452	4 433 152	4 101 093	1 802 948
	150	3 677	14 715 853	10 482 257	3 885 772
	200	3 253	36 076 194	18 652 471	6 343 447
	250	3 015	72 716 875	27 662 494	8 978 786

(a)

SIGN

Test	Valeur	nbEx	cfg_1	cfg_2	cfg_3
Variation oc_{max}	20	0	1 759 221	68 818	46 036
	60	0	1 759 221	529 047	249 263
	100	21	1 759 221	966 929	389 706
	140	286	1 759 221	1 402 336	526 987
	180	1 759	1 759 221	1 676 174	637 482
Variation $minconf$	0,1	1 713	1 759 221	1 192 583	646 080
	0,3	407	1 759 221	1 192 583	517 664
	0,5	253	1 759 221	1 192 583	480 855
	0,7	95	1 759 221	1 192 583	457 751
	0,9	88	1 759 221	1 192 583	449 788
Variation tw_{max}	50	27	76 380	76 171	45 052
	100	171	470 635	436 923	198 400
	150	95	1 759 221	1 192 583	457 751
	200	60	5 147 279	2 245 580	783 877
	250	43	12 730 926	3 503 690	1 154 581

(b)

FIGURE 4.5 – Nombre de règles extraites et générées par différentes configurations de TWINCLE sur les logs BIBLE et SIGN.

KOSARAK

Test	Valeur	nbEx	cfg_1	cfg_2	cfg_3
(1) Variation oc_{max}	20	1	1 630 702	201 171	185 647
	60	79	1 630 702	502 116	299 125
	100	691	1 630 702	888 066	454 182
	140	3811	1 630 702	1 264 835	618 556
	180	14 554	1 630 702	1 520 642	733 660
(2) Variation $minconf$	0,1	5 882	1 630 702	1 083 515	838 294
	0,3	3 854	1 630 702	1 083 515	683 915
	0,5	3 073	1 630 702	1 083 515	616 770
	0,9	1 581	1 630 702	1 083 515	487 239
	0,7	1 689	1 630 702	1 083 515	538 717
(3) Variation tw_{max}	50	403	113 727	113 386	83 727
	100	1 811	515 425	472 602	265 995
	150	1 689	1 630 702	1 083 515	538 717
	200	1 661	4 242 012	1 830 875	853 382
	250	1 646	9 677 266	2 651 971	1 178 120

(a)

FIFA

Test	Valeur	nbEx	cfg_1	cfg_2	cfg_3
(1) Variation oc_{max}	20	0	35 915 219	431 120	302 978
	60	8	35 915 219	4 343 963	1 672 363
	100	134	35 915 219	12 477 965	3 862 031
	140	1 608	35 915 219	22 824 630	6 335 069
	180	13 267	35 915 219	31 507 482	8 500 945
(2) Variation $minconf$	0,1	6 323	35 915 219	17 533 770	8 465 785
	0,3	2 555	35 915 219	17 533 770	6 017 949
	0,5	1 585	35 915 219	17 533 770	5 285 666
	0,7	493	35 915 219	17 533 770	5 082 161
	0,9	468	35 915 219	17 533 770	5 060 003
(3) Variation tw_{max}	50	258	888 427	876 322	418 920
	100	1 038	7 589 414	6 210 709	2 139 937
	150	493	35 915 219	17 533 770	5 082 161
	200	486		32 660 308	8 710 728
	250	478			12 650 505

(b)

FIGURE 4.6 – Nombre de règles extraites et générées par différentes configurations de TWINCLE sur les logs KOSARAK et FIFA.

4.2 Applications de l'instanciation PRISM

Dans cette section, nous expérimentons PRISM sur deux applications. La première est menée à partir d'un event log artificiel mais représentatif d'un processus réel. La seconde résulte d'une collaboration avec l'Agence Régionale de Santé (ARS) de l'ex région Auvergne, et porte sur un log réel construit à partir de données récoltées au Centre Hospitalier du Pays de Craponne-sur-Arzon (CHPCA) dans le département de la Haute-Loire (43). Cette double étude a pour but de tester l'efficacité de PRISM sur un processus structuré, dit "en lasagne", et sur un processus non structuré, dit "en spaghetti".

Une leçon de cuisine...

Lorsqu'il s'agit de modéliser un processus, il existe essentiellement deux types de processus : les processus dits "en lasagne" et les processus dits "en spaghetti". Les processus "en lasagne" ont une structure claire et bien définie. Il y a généralement très peu d'exceptions et le modèle est assez simple à comprendre pour les différentes parties prenantes. Il n'y a pas de définition formelle sur les caractéristiques d'un processus "en lasagne" ; cependant il est d'usage de dire qu'un "processus est "en lasagne" s'il est possible de s'accorder dessus assez facilement et qu'il représente au moins 80% des observations" [van der Aalst, 2016].
A l'inverse, les processus "en spaghetti" sont des processus très peu structurés, le modèle construit pouvant devenir très difficile à lire et interpréter. Cela est principalement dû à un fort degré de variabilité du processus modélisé, les différentes traces étant très différentes les unes des autres et ne formant pas de groupes homogènes. Les domaines comme celui de la santé où l'aléatoire a une place prépondérante, possèdent des processus qui ont de grandes chances d'être très peu structurés.
Souvent, les techniques ont des résultats et des performances différentes selon si elles considèrent un processus "en lasagne" ou un processus "en spaghetti".

4.2.1 Event log artificiel

Cette première étude de cas a pour but d'analyser les résultats obtenus par PRISM sur un processus structuré. Pour cela, nous avons choisi de nous inspirer d'un event log artificiel généré et mis à disposition par le Process Mining Group de l'Université Technique d'Eindhoven. Cet event log est représentatif d'un processus réel dans le sens où aujourd'hui, il existe beaucoup de procédures qui sont maîtrisées, et dont l'exécution laisse peu de place à l'aléa comme c'est le cas pour la mise en place de prothèses de hanche par exemple.

L'event log, que nous appelons "ARTIFICIEL", représente différentes prises en charge par un établissement de santé. Pour une prise en charge, plusieurs actes sont effectués ; e.g., prise de sang, scanner ; ainsi que plusieurs consultations médicales. Enfin, selon les résultats des différents diagnostics, le patient est soit amené à subir une opération chirurgicale, soit une ordonnance médicale lui est simplement prescrite. En plus de l'activité et du timestamp, chaque event possède un attribut "ressource". Cet attribut représente la ressource ayant exécuté l'activité. A partir de la valeur de cet attribut, nous proposons une nomenclature pour attribuer un coût à chaque event. Nous répartissons les 13 ressources existantes en 6 groupes : 4 internes en médecine, 3 médecins expérimentés, 2 infirmiers,

code	**1** 1re consultation	**2** Prise de sang	**3** Tests physiques	**4** 2e consultation	**5** Opération	**6** 3e consultation	**7** Scanner	**8** Prescription
Internes	1	2	2	2	3	3	2	2
Médecins experts	2	3	3	1	1	1	3	3
Infirmiers	2	2	1	3	✗	3	2	3
Techniciens	3	1	3	3	✗	3	3	✗
Pharmaciens	2	3	3	3	✗	3	3	1
Manipulateurs	3	3	3	3	✗	3	1	✗

TABLEAU 4.3 – Matrice de compatibilité ressource/activité.

2 manipulateurs, 1 pharmacien et 1 technicien. Pour des raisons de qualité de prises en charge et de coût financier, nous considérons que certaines ressources sont plus adaptées que d'autres pour exécuter certaines activités. Nous présentons dans la Figure 4.3 une matrice de compatibilité ressource/activité créée arbitrairement, qui définit le degré de compatibilité d'une ressource avec une activité (plus la valeur est haute moins la ressource et l'activité sont compatibles). Nous noterons une totale incompatibilité entre la ressource et l'activité, exprimée par le symbole ✗, et qui sera représenté par un coût très grand.

Le log que nous utilisons est composé de 1000 traces et 8 activités distinctes. En moyenne, chaque trace est composée de 7 events, la trace la plus courte et la plus longue étant composées respectivement de 6 et 7 events.

L'analyse que nous voulons effectuer sur ce log est de voir dans quelle mesure certaines activités peuvent être dépendantes les unes des autres à partir du moment où elles sont exécutées par les ressources adaptées.

Nous présentons le diagramme BPMN généré par l'Inductive Miner et son process tree équivalent dans les Figures 4.7a et 4.7b. Il est facile de constater que le processus est bien "en lasagne". La transformation du process tree en process tree minimal est présentée dans la Figure 4.7c. Toutes les activités de consultation étant des enfants d'un noeud-opérateur de type Séquence, il est possible de les supprimer, et ainsi laisser dans le process tree minimal les 3 activités en parallèle et les 2 activités dans une situation de choix exclusif, soit une réduction de 37,5% de l'espace de recherche. Pour $PT = \{N, r, m, c\}$ le process tree original, $PT_{min} = \{N \backslash \{n_2, n_7, n_{11}\}, r, m, c\}$.

Soit cfg_1 la fouille du log ARTIFICIEL sans PRISM ; i.e., en utilisant le process tree construit par l'Inductive Miner sans technique d'élagage de l'espace de recherche, et cfg_2 la fouille du log ARTIFICIEL avec PRISM ; i.e., en utilisant le process tree minimal et les techniques d'élagage de l'espace de recherche. Les résultats de ces deux configurations sont présentés dans le Tableau 4.8, dans lequel cpu, ram, $nbEx$ et $nbGen$ représentent respectivement le temps d'exécution (en ms), la consommation mémoire (en Mo), le nombre de règles extraites et le nombre de règles générées. La ligne "ratio" représente le rapport des différentes mesures évaluées sur cfg_2 sur celles de cfg_1.

Les paramètres utilisés pour effectuer ces expérimentations sont présentés dans le Tableau 4.8b. Le coût global maximum oc_{max} est fixé à 10 pour que les techniques d'élagage soient efficaces tout en permettant l'extraction d'un nombre de règles suffisant. Pour

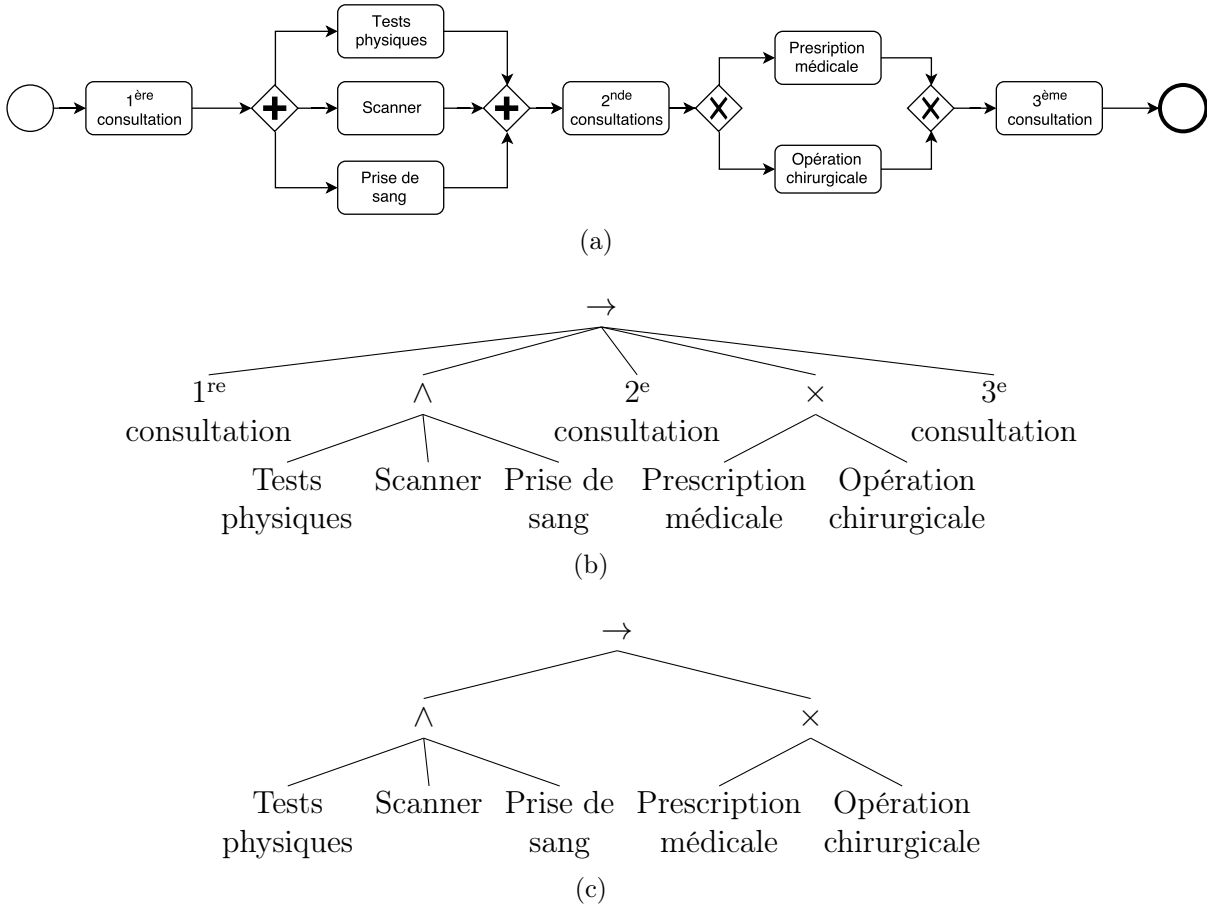


FIGURE 4.7 – *(a)* Diagramme BPMN, *(b)* process tree généré par l'Inductive Miner sur le log ARTIFICIEL et *(c)* le process tree minimal.

que les règles extraites représentent des dépendances fortes et pertinentes, nous avons fixé le seuil de confiance minimum $minconf$ à 0,7. Avec $minconf$ au-dessus de 0,7, les dépendances se rapprocheraient exclusivement de dépendances totales ; au dessous elles ne seraient plus "fortes", et perdraient donc leur intérêt. Nous avons fixé la fenêtre de temps à tw_{max} à 10 jours, signifiant que les dépendances perdent de leur pertinence d'un point de vue métier si l'exécution des activités concernées ne se produit pas dans une fenêtre de 10 jours. Le seuil de cohérence maximum coh_{max} est fixé à 0,4 ; ce qui signifie que même si le processus semble être en lasagne ; i.e., maîtrisé en termes de control-flow ; nous autorisons la variation du coût global d'une dépendance dans les différentes traces qui la contiennent à représenter 40% du coût global de la dépendance dans le log. Nous avons fixé k à $\frac{1}{24}$, signifiant que l'on juge préférable de confier l'exécution d'une activité à une ressource moins qualifiée pour réduire la durée de séjour du patient de 24 heures. Les paramètres α et β sont fixés à 0,5 pour que l'importance de la variation de la durée des séjours soit identique à celle des coûts dans le calcul de la cohérence.

Le process tree minimal ne possède plus que 5 activités, contre 8 dans le process tree original. Nous remarquons que l'utilisation de PRISM a permis une réduction non négligeable de la fouille du log dans son ensemble. Le nombre de règles générées et extraites par cfg_2 représente respectivement 6,82% et 28,9% du nombre de règles générées et extraites par cfg_1. Le temps d'exécution et la consommation mémoire de cfg_2 représentent respectivement 22,6% et 51,1% du temps d'exécution et de la consommation mémoire de cfg_1.

oc_{max}	$minconf$	$tw_{max}(jours)$	coh_{max}	k	α	β
10	0,7	10	0,4	1/24	0,5	0,5

(a)

	cpu (ms)	ram (Mo)	nbGen	nbEx
cfg_1	358	77,7	586	38
cfg_2	81	39,7	40	11
$ratio$	22,6%	51,1%	6,82%	28,9%

(b)

FIGURE 4.8 – *(a)* Paramètres utilisés pour les expérimentations sur le log ARTIFICIEL et *(b)* les résultats d'exécution avec et sans PRISM.

r	$oc(r, \mathcal{L})$	$conf_{\mathcal{L}}(r)$	$coh(r, \mathcal{L})$	$sup_{\mathcal{L}}(r)$
{7,3}→{8}	6,2	0,87	0,26	780
{7,3,2}→{8}	7,0	0,86	0,20	240
{2,7,3}→{8}	9,1	0,78	0,28	280

TABLEAU 4.4 – Exemples de règles extraites par PRISM sur le log ARTIFICIEL.

Dans le Tableau 4.4, nous présentons trois règles extraites qui nous paraissent particulièrement pertinentes. Pour des raisons de lisibilité, nous représentons les activités concernées par le code de chaque activité. L'analyse principale que nous allons pouvoir effectuer sur ce processus est de voir dans quelle mesure l'ordre d'exécution des activités en parallèle influe sur l'activité exécutée parmi celles contenues dans la situation de choix exclusif et le temps pris pour l'exécuter.

La première règle {7,3}→{8} indique que lorsque le patient effectue un scanner puis des tests physiques, il se voit simplement prescrire des médicaments dans 87% des cas dans la fenêtre de temps prédéfinie.

Il est intéressant de voir comment l'exécution de l'activité "Prise de sang" influe sur le déroulement du processus. La corrélation présentée dans la première règle change peu dans la deuxième règle {7,3,2}→{8}, lorsque la prise de sang est effectuée après le scanner et les tests physiques. Le coût global de la règle augmente légèrement, mais sa confiance reste quasiment identique, à 0,86. Il est à noter que la cohérence de la deuxième règle est meilleure que la première, ce qui semble indiquer que lorsque la prise de sang est effectuée après le scanner et les tests physiques, le déroulement de la suite du processus est mieux maîtrisé en termes de temps et de ressources.

Il est possible d'appuyer cette hypothèse avec la troisième règle {2,7,3}→{8} dans laquelle les tests physiques sont toujours effectués après le scanner, mais où la prise de sang est effectuée tout au début. Le coût global de la règle a été plus impacté que pour la deuxième règle, et sa confiance, même si elle reste correcte, a diminué, passant à 0,78. De plus, la cohérence par rapport à la deuxième règle est inférieure de 8 points de pourcentage, indiquant qu'effectuer la prise de sang au début a un impact sur les ressources effectuant les autres activités ou le temps pris pour les effectuer.

Il est aussi à noter que parmi les règles extraites, la quasi totalité avait l'activité "Prescription" dans le conséquent, indiquant que le choix d'effectuer une opération chirurgicale est peut-être plus soumis à l'aléa et moins maîtrisé en termes de temps, impliquant qu'aucune des principales règles extraites ne concerne l'activité "opération chirurgicale".

Nous avons montré à travers cette première étude de cas qu'il était possible d'extraire des connaissances intéressantes sous la forme de dépendances dans un processus "en lasagne". Dans la seconde étude de cas, nous allons évaluer dans quelle mesure il est possible d'effectuer le même type d'analyse sur un processus non structuré, "en spaghetti".

4.2.2 Le Centre Hospitalier du Pays de Craponne-sur-Arzon (CHPCA)

Cette étude est le résultat d'une collaboration entre le CHPCA, le Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS CNRS UMR 6158) de Clermont-Ferrand, le Centre de Recherche Clermontois en Gestion et Management (CRCGM, EA 3849) et l'Agence Régionale de Santé (ARS) de l'ex région Auvergne. Le CHPCA est un établissement public de santé d'une capacité de 154 lits. Il couvre l'ensemble de la filière gériatrique avec une activité de Médecine-Chirurgie-Obstétrique (MCO) et de Soins de Suite et de Réadaptation (SSR). Il est composé d'un Établissement d'Hébergement pour Personnes Âgées Dépendantes (EHPAD) dont une Unité d'Hébergement Renforcée (UHR), et d'un Accueil de Jour.

activité	code	activité	code
Start	1	End	5
MCO	3	SSR	13
Domicile IN	9	Domicile OUT	7
EHPAD CHPCA IN	14	EHPAD CHPCA OUT	15
EHPAD Autre IN	2	EHPAD Autre OUT	4
MCO Autre IN	6	MCO Autre OUT	8
SSR Autre IN	16	SSR Autre OUT	17
Traitement Externe	10	Décès	12

Tableau 4.5 – Liste des activités présentes dans le log CHPCA.

L'objectif de cette étude est de tenter de dégager des tendances à partir des séjours enregistrés dans le Système d'Information du CHPCA. A partir des données disponibles, nous avons construit le log comme suit. Pour un patient accueilli au CHPCA, un séjour commence par une activité correspondant à sa structure d'origine ; i.e., celle qui l'a transféré au CHPCA. Ce même séjour se termine par une activité représentant la structure de destination ; i.e., celle dans laquelle il est admis après être sorti du CHPCA. Nous noterons la présence des structures "Domicile" et "Décès", représentant respectivement le fait que le patient n'était pas pris en charge avant d'être accueilli au CHPCA, et le fait que le patient décède alors qu'il était pris en charge par le CHPCA. Au sein du CHPCA, il peut être pris en charge soit par le service de MCO, soit par le service de SSR, soit les deux. Dans ce dernier cas, la prise en charge en MCO est effectuée avant la prise en charge en SSR. Chacun de ces services fait donc l'objet d'une activité. Une trace est la concaténation des séjours d'un patient de 75 ans ou plus entre janvier 2013 et décembre 2015. Si un patient a effectué plus d'un séjour sur cette période, une activité "Traitement Externe" est ajoutée entre deux séjours. Comme il n'est pas possible de savoir si le patient a reçu un suivi entre les deux séjours, cette activité représente une potentielle prise en charge externe. De plus, le timestamp exact de chaque event n'est pas disponible, le niveau d'abstraction le plus bas auquel nous ayons accès étant le numéro de semaine. Parce que nous n'avons pas constaté deux séjours consécutifs la même semaine pour un même

patient, le jour des admissions a été arbitrairement fixé au lundi et le jour des sorties à été fixé au vendredi. Dans la suite de cette étude, nous baserons nos interprétations sur ces timestamps. Enfin, pour des raisons de structuration, deux activités "Start" et "End" sont ajoutés respectivement au début et à la fin de chaque trace.

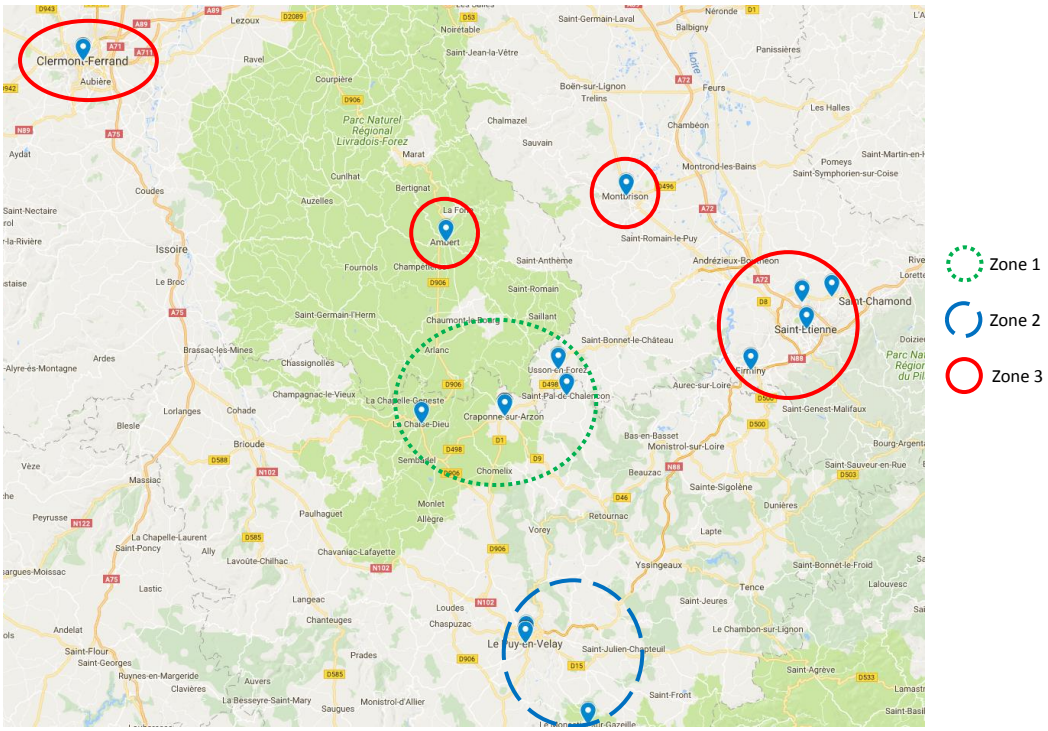
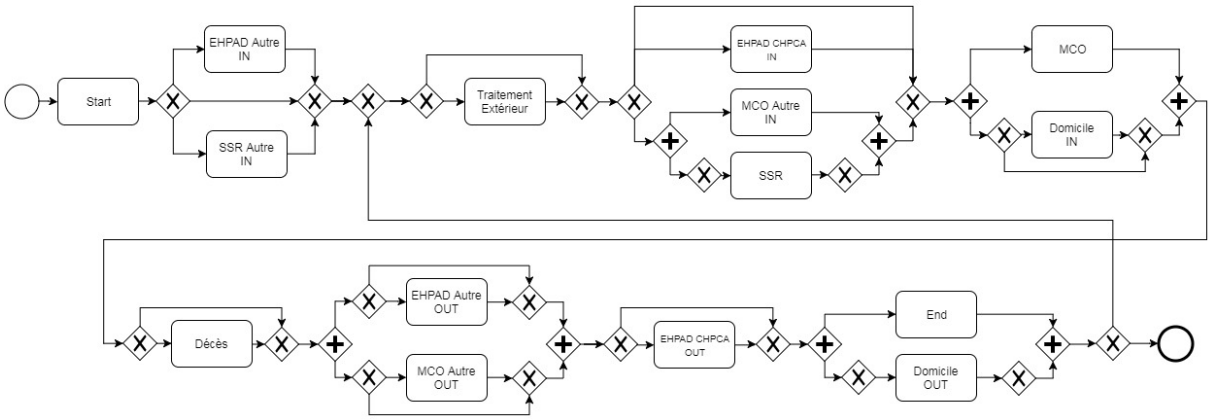


FIGURE 4.9 – Carte des zones selon le lieu d'implantation des structures d'origine et de destination des patients.

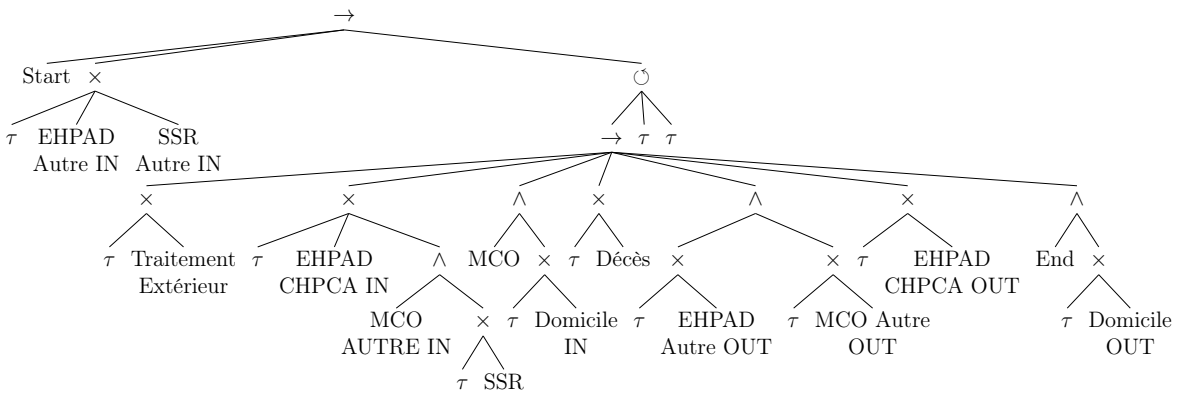
Le CHPCA a pour vocation d'être un établissement de proximité ; sa volonté est donc de favoriser les transferts dans des zones proches pour que les patients gardent un contact avec leur environnement social. De plus, bien qu'un retour à Domicile est souhaité de préférence, le CHPCA tente de valoriser sa *filière intégrée* pour un meilleur suivi des patients. La filière intégrée met l'accent sur un suivi "interne", via un hébergement dans l'EHPAD du centre. Pour être en accord avec ces choix, nous définissons le coût d'un event comme la proximité des structures d'origine et de destination par rapport au CHPCA. La proximité se traduit à la fois par la proximité géographique mais aussi les relations existantes, souvent informelles entre établissements. Dans la Figure 4.9, nous présentons une carte annotée des lieux d'implantation des différentes structures, ainsi que leur coût associé, sur une échelle de 1 à 3 en fonction de leur zone (cf carte) ; 1 étant la zone la plus proche du CHPCA. Sur les différents séjours, certains représentent des prises en charge de patients "de passage", avec des transferts vers des structures assez éloignées ; e.g. La Ciotat ou Montélimar. Ces séjours ont donc été filtrés. Afin d'interpréter au mieux l'extraction de règles qui sera effectuée, les séjours "chroniques" ; e.g., transfusion, ont aussi été filtrés.

Au total, 968 séjours ont été extraits sur les trois années, et représentent plus de 80% du nombre de séjours total sur cette période. Le log contient 602 traces et 16 activités distinctes présentées dans le Tableau 4.5, le suffixe "IN" représentant la structure d'origine et le suffixe "OUT" représentant la structure de destination. Comme nous modifions le label des activités pour distinguer les différentes occurrences d'une même activité, nous

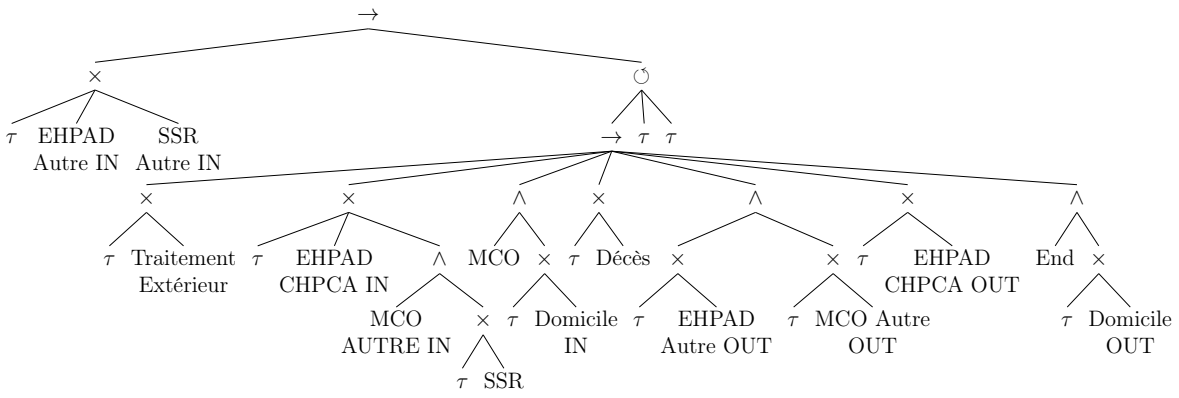
avons 109 activités au total. En moyenne chaque trace est composée de 7 events, la trace la plus courte et la plus longue étant composées respectivement de 4 et 18 events.



(a)



(b)



(c)

FIGURE 4.10 – *(a)* Diagramme BPMN, *(b)* process tree généré par l'Inductive Miner sur le log du CHPCA et *(c)* le process tree minimal.

Nous présentons dans la Figure 4.10a le modèle généré avec l'Inductive Miner. Comme nous pouvons le voir, le modèle construit n'est pas aussi simple que ce à quoi nous pourrions nous attendre. Deux raisons peuvent l'expliquer. La première est que le processus est peu structuré ; i.e., il est composé de beaucoup de variantes, qui impliquent un grand nombre de situations de choix. La seconde raison est que l'Inductive Miner a tendance à

générer des modèles avec une grande généralisation ; i.e., qui autorisent beaucoup plus de comportements que ceux observés dans le log.

La conséquence directe de cette faible structuration du processus est que l'étape #2 de la méthodologie KITE relative à la minimisation du process tree, voit son efficacité diminuée. Le process tree de ce processus est présenté dans la Figure 4.10b et est équivalent au diagramme BPMN de la Figure 4.10a. A première vue, le process tree est plus large que profond, ce qui semble être une bonne chose. En effet, une grande largeur réduit les risques d'avoir des opérateurs de type XOR imbriqués, synonymes d'absence de dépendances. De plus, le nœud n_8 est un opérateur de type Séquence possédant beaucoup d'enfants, cela favorise donc l'existence de dépendances. Ces caractéristiques, qui sont propices à la détection de dépendances, empêche une minimisation efficace du process tree. Si l'on applique le cycle de sélection/réduction présenté dans la sous-section 3.3.2, seule l'activité "Start" sera supprimée ce qui représente à peine 5,6% des activités. Pour $PT = \{N, r, m, c\}$ le process tree original, $PT_{min} = \{N \backslash \{n_2\}, r, m, c\}$.

oc_{max}	$minconf$	$tw_{max}(jours)$	coh_{max}	k	α	β
8	0,7	30	0,25	1/720	0,5	0,5

(a)

	cpu (ms)	ram (Mo)	nbEx	nbGen
cfg_1	397	64	4 212	61 425
cfg_2	255	51	4 212	21 521
$ratio$	64,2%	79,7%	100%	34,9%

(b)

FIGURE 4.11 – *(a)* Paramètres utilisés pour les expérimentations sur le log CHPCA et *(b)* les résultats d'exécution avec et sans PRISM.

Soit cfg_1 la fouille du log CHPCA sans PRISM ; i.e., sans minimalisation du process tree ni techniques d'élagage de l'espace de recherche, et cfg_2 la fouille du log CHPCA avec PRISM ; i.e., avec minimisation du process tree et techniques d'élagage de l'espace de recherche. Les résultats de ces deux configurations sont présentés dans le Tableau 4.11b, dans lequel cpu, ram, $nbEx$ et $nbGen$ représentent respectivement le temps d'exécution (en ms), la consommation mémoire (en Mo), le nombre de règles extraites et le nombre de règles générées. La ligne "ratio" représente le rapport des différentes mesures évaluées sur cfg_2 sur celles de cfg_1.

Les paramètres utilisés pour effectuer ces expérimentations sont présentés dans le Tableau 4.11a. Le coût global maximum oc_{max} est fixé à 8 pour que les techniques d'élagage soient efficaces tout en permettant l'extraction d'un nombre de règles suffisant. Pour les même raisons que la première étude de cas, nous avons fixé le seuil de confiance minimum $minconf$ à 0,7. Nous avons fixé la fenêtre de temps à tw_{max} à 30 jours, signifiant que les dépendances perdent leur pertinence d'un point de vue métier si l'exécution des activités concernées ne se produit pas dans une fenêtre d'un mois. Le seuil de cohérence maximum coh_{max} est fixé à 0,25 ; ce qui signifie que nous autorisons la variation du coût global d'une dépendance dans les différentes traces qui la contiennent à représenter au plus un quart du coût global de la dépendance dans le log. Nous avons fixé k à $\frac{1}{720}$, signifiant que l'on juge préférable de garder un patient 1 semaine (720 heures) de plus plutôt que de le transférer dans une structure éloignée de son environnement social. Les paramètres α et β sont fixés

à 0,5 pour que l'importance de la variation de la durée des séjours soit identique à celle des coûts dans le calcul de la cohérence.

r	$oc(r, \mathcal{L})$	$conf_{\mathcal{L}}(r)$	$coh(r, \mathcal{L})$	$sup_{\mathcal{L}}(r)$
{14, 3}→{15}	4,21	0,71	0.21	34
{2, 3}→{4}	4,65	0,75	0,24	24
{#2_10, #3_9}→{#3_3, #3_7}	5,37	0,71	0,24	10

TABLEAU 4.6 – Exemples de règles extraites par PRISM sur le log CHPCA.

Dans le Tableau 4.6, nous présentons trois règles extraites qui nous paraissent particulièrement pertinentes. Pour des raisons de lisibilité, nous représentons les activités concernées par le code de chaque activité, tel qu'indiqué dans le Tableau 4.5. En plus de fournir des informations sur les activités qui ont une dépendance, elles vont aussi permettre d'atténuer la sur-généralisation du processus modélisé.

La première règle indique que 71% des patients venant de l'EHPAD du CHPCA et pris en charge par l'unité de MCO ont pu retourner dans ce même EHPAD par la suite. Cette confiance élevée semble montrer que le CHPCA arrive à mettre en place une filière intégrée pertinente.

De la même manière, la seconde règle indique que 75% des patients provenant d'un EHPAD alentour et pris en charge en MCO retourne dans un EHPAD alentour, qui semble être a priori le même. Une fois de plus, cela montre l'effort fourni par le CHPCA pour conserver les patients dans un environnement social de proximité. Toutefois, des tests supplémentaires que nous avons effectués nous ont permis de constater que ces deux règles ne sont plus extraites dès lors que la fenêtre de temps autorisée est définie en dessous de 25 jours. Nous pouvons en conclure que la volonté de favoriser les transferts de patients dans un environnement proche du CHPCA (via la filière intégrée ou un retour à domicile), se fait au détriment de la durée de la prise en charge.

Enfin, la troisième règle est un peu plus complexe mais tout de même très intéressante. La notation $\#i\_a$ représente la i^e occurrence de l'activité a. Par exemple, l'antécédent de la troisième règle est composé de la 2^e occurrence de l'activité "Traitement Extérieur" et de la 3^e occurrence de l'activité "Domicile IN". Cette règle peut donc s'interpréter comme "les patients admis pour la 3^e fois au CHPCA, provenant de leur domicile pour chaque séjour, ont pour 71% été pris en charge par le service de MCO et renvoyé à leur domicile. Cette règle vient appuyer les deux premières en démontrant que cet effort de conserver le patient dans un contexte social favorable se poursuit sur les séjours récurrents.

Il ne faut pas oublier que le seuil de cohérence fixé prouve d'autant plus que les transferts de patients effectués par le CHPCA se font très souvent dans un environnement de proximité et dans des temps relativement similaires.

Nous avons montré à travers cette seconde étude de cas qu'il était possible d'extraire des connaissances intéressantes sous la forme de dépendances dans un processus "en spaghetti". De plus, nous constatons que les limitations rencontrées lors des benchmarks en termes de consommation mémoire ne s'appliquent pas dans les deux études de cas menées. Toutefois, nous remarquons que la phase de minimisation du process tree est moins efficace lorsque le processus est "en spaghetti". En effet, dans la seconde étude de cas, seulement 5,6% du process tree initial a été élagué, contre 37,5% pour celui de la première étude de cas.

Conclusion

Dans ce chapitre nous avons expérimenté les différentes approches proposées dans le chapitre précédent.

Nous avons montré que l'algorithme TWINCLE répondait aux objectifs fixés : d'une part il permet l'extraction de règles séquentielles à bas coût, d'autre part les optimisations que nous avons proposées permettent de réduire les temps de calcul de manière conséquente.

Ensuite, nous avons tester l'instanciation PRISM sur deux applications. Dans la première, le processus utilisé était structuré, "en lasagne". Nous avons montré que ce type de processus permettait de réduire efficacement l'espace de recherche intéressant à explorer. Dans la seconde, nous avons utilisé un processus très peu structuré, en "spaghetti". Nous avons constaté que les caractéristiques de ce type de processus impactent négativement la capacité de PRISM de réduire l'espace de recherche, mais permet quand même d'extraire des dépendances intéressantes.

Conclusion de Partie

L'objectif de cette première partie était double. Dans un premier temps, nous devions répondre à la problématique qui consiste à détecter des dépendances entre activités dans un processus. Dans un second temps, nous devions répondre à la problématique qui consiste à définir un nouveau référentiel pour évaluer l'importance relative de ces dépendances.

Afin de répondre à ces problématiques, nous avons choisi de nous tourner vers le process mining, une discipline à mi-chemin entre la modélisation de processus et la fouille de données.

Dans le chapitre 2, nous avons présenté un aperçu des différentes analyses qu'il est possible de faire avec le process mining. Nous avons ensuite fait un état des lieux des applications qui ont été faites dans le domaine de la santé. Nous avons montré que la santé est un domaine très intéressant d'un point de vue académique et que de nombreux challenges sont encore à relever. Cependant, nous avons pu constater que les techniques existantes ne sont pas capables de répondre aux problématiques que nous avons proposées.

	Relations de dépendance						
Technique	bi./uni.	tot./fort.	dir./ind.	sim./mult.	ch./ord.	réf.	sim.
α-miner						fr.	
Fuzzy miner						fr.	✓
α^{++}miner			✓			fr.	
Heuristics miner	✓	✓	✓			fr.	
IDM	✓	✓	✓			fr.	✓
PRISM	✓	✓	✓	✓	✓	ut	✓

TABLEAU 4.7 – Récapitulatif des limites de l'existant en Process Mining et positionnement de nos travaux.

Pour pallier ces limites, notre contribution dans le chapitre 3 a été double. Dans un premier temps, nous avons présenté la méthodologie que nous avons développée pour utiliser le pouvoir d'expression des modèles de processus afin de (1) définir une importance relative aux différentes données disponibles et (2) de réduire le temps de calcul nécessaire pour extraire des connaissances intéressantes à partir de ces données. Cette méthodologie, que nous appelons *KITE*, a pour objectif d'être générique, indépendante du domaine d'application et de la technique d'extraction de connaissances utilisée.

Afin de répondre à nos problématiques, nous avons ensuite proposé deux instanciation. La première, que nous appelons *IDM*, a été développée dans l'objectif d'évaluer l'impact des choix dans l'exécution d'un processus. Nous avons ensuite proposé une seconde instanciation pour étendre le périmètre d'application d'IDM. Cette instanciation, que nous appelons *PRISM*, a pour objectif d'évaluer l'impact des choix et de leur ordre sur l'exécution d'un processus. Après avoir construit un modèle répondant aux critères fixés, PRISM effectue un *cycle de sélection/réduction*, pour élaguer l'espace de recherche à explorer. L'étape d'extraction de connaissances se fait via l'algorithme *TWINCLE*. Pour extraire des dépendances intéressantes entre les activités d'un processus de santé, nous avons proposé et formalisé le problème d'extraction de règles séquentielles à bas coût. TWINCLE est un algorithme que nous avons spécialement développé pour adresser ce problème.

Dans le chapitre 4, nous avons effectué différentes expérimentations afin d'évaluer les performances des méthodes proposées. Nous avons montré sur des jeux de données issus de la littérature que TWINCLE était capable d'extraire des règles séquentielles à bas coût, tout en tirant profit des optimisations que nous avons proposées pour réduire les temps de calculs. Ensuite, à travers deux applications, nous avons montré que les performances de PRISM en termes de temps de calcul et de qualité de dépendances extraites sont en accord avec les objectifs que nous nous sommes fixés pour répondre à nos problématiques. Nous positionnons dans le Tableau 4.7 les instanciations IDM et PRISM par rapport aux autres techniques de process mining disponibles dans leur capacité à extraire les différents types de dépendances que nous avons définis. Nous constatons que PRISM est bien capable de détecter tout type de dépendance et ce, avec un référentiel "utile".

Cependant, nous rencontrons aussi plusieurs limites qui impactent l'applicabilité et l'efficacité de l'instanciation PRISM. Dans cette section, nous mettons en évidence les principales limites tout en présentant quelques pistes d'amélioration.

Dans l'étape #1 relative à la construction du modèle, la principale limite est dans la technique utilisée. Dans PRISM, nous avons décidé d'utiliser l'Inductive Miner car cette technique répondait aux différents prérequis que nous nous étions imposés. Cependant, comme toute technique, l'Inductive Miner a elle aussi des limites. Par exemple, elle a tendance à la sur-généralisation ; i.e., les modèles construits autorisent beaucoup de comportements non observés dans le log. Cette sur-généralisation se traduit souvent par la présence de nombreuses situations de choix. Bien que cela n'empêche pas la détection de dépendances entre les activités, un modèle dont la généralisation est élevée va impacter l'efficacité de l'étape #2. Une piste d'amélioration est de changer de technique, et par extension de structure algorithmique, pour mieux réduire l'espace de recherche.

L'étape #2, qui vise à réduire la représentation du modèle en process tree, voit son applicabilité réduite si certaines caractéristiques sont présentes. Pour les règles prédéfinies pour effectuer cette réduction, la présence de nombreuses situations de choix réduit leur applicabilité. De plus, si une situation de parallélisme, représentée par un nœud-opérateur AND, se situe près du nœud-racine, alors il n'est quasiment plus possible de réduire le process tree. Le fait que nous voulons analyser l'impact des choix et de l'ordre de ces choix est le principal responsable de cette limite. Une alternative pour limiter l'impact de la structure du process tree est d'effectuer la minimisation dans l'objectif soit d'analyser l'impact des choix, soit d'analyser l'impact de l'ordre des choix ; mais pas les deux. De cette manière il serait possible de relâcher les contraintes fixées par les règles de sélection dans la phase de sélection et nous pourrions supprimer plus d'activités. Par exemple, une situation de choix exclusif n'est pas intéressante si l'on se concentre sur l'impact de l'ordre

des choix, de la même manière qu'une situation de parallélisme n'a pas d'importance si nous nous concentrons uniquement sur les choix et non leur ordre. Une autre piste de réflexion que nous proposons est de développer une méthode pour analyser a priori comment les différentes caractéristiques d'un event log influencent l'efficacité du processus de minimisation.

Ensuite, l'étape #3 vise à effectuer la fouille du log. Bien que l'élaboration de l'algorithme TWINCLE réponde à plusieurs problématiques théoriques et pratiques, plusieurs améliorations sont à apporter. Tout d'abord, les mesures utilisées ont été sélectionnées pour répondre aux objectifs fixés. Bien évidemment, pour des raisons de complexité, il est impossible d'intégrer toutes les mesures que l'on jugerait utile dans le même algorithme. Une piste intéressante serait de voir dans quelle mesure il est possible d'ajouter une ou plusieurs mesures pour compléter celles déjà présentes et contribuer à l'extraction de règles encore plus pertinentes. Au niveau des performance de TWINCLE, la consommation mémoire est un axe d'amélioration. Bien que dans certaines configurations cette consommation reste acceptable, il serait nécessaire de la réduire. Nous sommes convaincus que cette réduction est faisable sans changer l'essence de l'algorithme ; et une piste envisageable est de modifier son implémentation pour faire usage de toutes les techniques d'optimisation disponibles. Une autre alternative serait de changer de structure d'évaluation. Dans nos travaux, nous nous sommes inspirés des *utility-table* pour calculer le support des règles, mais il serait intéressant de voir dans quelle mesure d'autres structures peuvent améliorer les performances de TWINCLE.

L'étape #4 vise à définir les résultats à obtenir pour permettre une analyse pertinente. Dans PRISM, nous avons décidé de représenter les dépendances entre activités sous la forme de règles. De manière générale, nous constatons aussi des limites à la capacité des différentes règles à répondre à certaines questions. Le pouvoir d'expression des règles en est la cause. Une règle est une séquence d'items, la connaissance qu'elles représentent est donc restreinte à n'être qu'une succession d'activités entre lesquelles une dépendance existe. De plus, la définition du concept générique d'utilité (que l'on appelle aussi "profit" ou "coût" selon les cas) est "fixe" ; i.e., l'utilité d'un event est seulement dépendante de l'event auquel il est associé. Cette définition empêche de répondre à certaines questions. Par exemple, dans le cas du log CHPCA, considérons que le "coût" d'un event est le niveau de dépendance du patient au moment de sa prise en charge. Une problématique qu'il serait intéressant d'étudier serait l'extraction des règles pour lesquelles les activités et l'ordre des activités de l'antécédent impactent une réduction de l'autonomie du patient pour les activités contenues dans le conséquent. Cette question implique que le coût d'un event n'est pas "fixé" mais plutôt "dynamique", car il dépendrait des activités qui le précèdent dans la règle.

Bien évidemment, nous ne pouvons traiter toutes les perspectives proposées. Par conséquent, nous choisissons de nous concentrer sur celles énoncées pour l'étape #4 de PRISM. Les motivations qui nous poussent à faire ce choix sont doubles. D'une part le pouvoir d'expression des règles séquentielles est très facilement "extensible", c'est l'objectif même des techniques de process mining. D'autre part nous voyons dans la prise en compte du concept d'utilité/profit lors de la construction de modèles de processus à la fois un challenge scientifique très intéressant et un apport important d'un point de vue des connaissances métier. Deux nouvelles problématiques émergent : représenter les dépendances entre activités sous forme de modèle, et la prise en compte de nouvelles caractéristiques pour mesurer l'importance d'une dépendance.

Dans la seconde partie de ce manuscrit, nous tentons de répondre à ces deux nouvelles problématiques.

Deuxième partie

Connaissances et Extraction de Modèles de Processus

Introduction de Partie

Dans la première partie de nos travaux, nous avons constaté que les performances des approches proposées sont diminuées lorsqu'il s'agit d'analyser des processus peu structurés. Dans la seconde partie du manuscrit, nous proposons une alternative à ces approches, dont l'objectif est double. D'une part, nous voulons "enrichir" le pouvoir d'expression des dépendances détectées. Dans la partie précédente, les dépendances étaient exprimées sous la forme de règles séquentielles. Dans l'optique d'extraire plus d'informations pertinentes, nous voulons autoriser plus de comportements que la séquence dans les relations entre activités dépendantes. D'autre part, nous voulons prendre en compte de nouvelles caractéristiques pour mesurer l'importance des dépendances.

Cette partie est organisée comme suit. Dans le Chapitre 5, nous présentons les modèles *locaux* et *déclaratifs*. Ces types de modèles sont des alternatives à la modélisation "globale" de processus. Ils ont prouvé pouvoir fournir des informations plus pertinentes que les modèles globaux lorsqu'il s'agit de processus peu structurés.

Dans le Chapitre 6, nous définissons le problème *d'extraction de modèles de processus partiels profitables*. Extension du problème d'extraction de modèles de processus partiels, ce nouveau problème prend en compte, lors de la construction des modèles, l'intérêt que porte un analyste sur certains aspects du processus. Nous appelons ces nouveaux modèles les *High-Utility Local Process Models* (HU-LPMs). Cette extension permet donc de pouvoir analyser un même event log selon différents points de vue métier. Nous proposons une méthode pour extraire l'ensemble des HU-LPMs. Pour palier des limitations en termes de temps de calcul, nous proposons par la suite des méthodes approchées pour effectuer l'extraction des HU-LPMs dans des temps raisonnables.

Dans le Chapitre 7, nous évaluons les performances des approches proposées dans cette seconde partie sur différents jeux de données issus de la littérature, générés ou extraits dans le cadre de nos travaux. Dans un premier temps, nous montrons comment les modèles de processus partiels profitables sont capables de gérer les processus peu structurés et de donner des informations pertinentes en fonction du point de vue abordé. Dans un second temps, nous testons dans quelle mesure les méthodes approchées proposées permettent de réduire les temps de calcul nécessaires à l'extraction des HU-LPMs. Nous évaluons aussi l'impact des caractéristiques d'un jeu de données sur l'efficacité des méthodes approchées.

Chapitre 5

Etat de l'Art

Sommaire

Introduction

Dans ce chapitre, nous traitons des travaux de la littérature en process mining relative à la modélisation de processus, et détaillons plus particulièrement les variantes qui se rapportent à notre problématique.

Les techniques de Process Discovery peuvent être classées selon différents critères. Certaines de ces techniques produisent des modèles de processus utilisant des notations avec une *sémantique formelle*, dans lesquels les comportements autorisés par le modèle sont formellement définis. A l'inverse, d'autres techniques construisent des modèles de processus qui ne possèdent pas ce formalisme. Conjointement, les techniques de process discovery peuvent aussi être classées en techniques *"globales"* (ou *"end-to-end"*) qui construisent des modèles représentant les observations présentes dans le log du début à la fin, et en techniques *"locales"* (ou *"pattern-based"*), qui produisent des modèles de processus décrivant les comportements observés dans le log seulement de manière partielle.

Technique	Sémantique formelle	Global/Local
Fuzzy Miner [Günther and van der Aalst, 2007]		Global
Language-based regions [Bergenthum et al., 2007]	✓	Global
ILP miner [van der Werf et al., 2008]	✓	Global
Declare Miner [Maggi et al., 2011]	✓	Global
Inductive Miner [Leemans et al., 2013]	✓	Global
SUBDUE patterns [Diamantini et al., 2013]		Local
Episode Miner [Leemans and van der Aalst, 2014]		Local
MINERful [Ciccio and Mecella, 2015]	✓	Global
LPM Discovery [Tax et al., 2016b]	✓	Local
Instance graphs [Diamantini et al., 2016b]		Local
Behavioral PM [Diamantini et al., 2016a]		Local
DPIL Miner [Schönig et al., 2016a]	✓	Global
TBDeclare Miner [Di Ciccio et al., 2016]	✓	Global
Split Miner [Augusto et al., 2017]	✓	Global

TABLEAU 5.1 – Classification d'une sélection de techniques de process discovery.

Il existe une pléthore de techniques de process discovery [van der Aalst, 2016]. Nous en présentons un aperçu représentatif dans le Tableau 5.1. Les techniques produisant un modèle avec une notation formelle sont annotées du symbole ✓. Les techniques sont aussi réparties en fonction de leur périmètre global ou local.

La majeure partie des techniques de process discovery génère des modèles globaux, mais surtout *procéduraux* (aussi appelés *impératifs*). Un modèle procédural spécifie explicitement l'ordre des activités du début à la fin du processus. Lorsque le processus est flexible ; i.e., il autorise beaucoup d'ordres entre les activités, les techniques de process discovery qui produisent des modèles de processus procéduraux tendent à produire soit des modèles avec une généralisation trop élevée autorisant trop de comportements, soit des modèles de processus "en spaghetti" ; i.e., des modèles très complexes et peu compréhensibles. Nous avons pu démontrer cela dans la première partie de ce manuscrit.

L'objectif de ce chapitre est de présenter des types de modèles alternatifs pour résoudre ce problème : les modèles *locaux* et les modèles *déclaratifs*. Contrairement aux modèles habituellement générés qui représentent les processus de bout en bout, les modèles locaux modélisent des fragments de comportements fréquents. Les modèles déclaratifs sont des

modèles dont l'exécution n'est pas explicite, mais encadrée par un ensemble de règles. Ces deux types de modèles font l'objet des deux premières sections de ce chapitre.

Parmi les différentes techniques présentées, certaines sont plus adaptées pour répondre à nos problématiques. Dans la section, 5.3, nous présentons la LPM Discovery, la technique de laquelle nous nous inspirons pour développer notre nouvelle approche.

5.1 Construction de modèles locaux

Les modèles locaux représentent une partie de processus. Dans la suite de cette section nous détaillons différentes façons de les représenter.

Local Process Model (LPM) Discovery [Tax et al., 2016b] est une technique qui construit de façon itérative un ensemble de modèles de processus locaux, les *Local Process Models* (LPMs) ; i.e., des fragments de processus décrivant des comportements fréquents. Ces modèles "partiels" ont un grand pouvoir d'expression, et sont capables de modéliser les choix, les parallélismes ou encore les boucles.

L'*Episode Miner* [Leemans and van der Aalst, 2014] est une technique à mi-chemin entre l'episode mining et le process mining. Cette technique extrait un ensemble d'*épisodes*, un épisode étant un ensemble d'events partiellement ordonnés, depuis un event log. Cependant, un épisode est un type de construction dont le pouvoir d'expression ne permet pas de modéliser les boucles ou les choix exclusifs. De plus, cette technique ne permet pas d'être facilement adaptée pour considérer ces comportements non pris en compte.

Après avoir transformé chaque trace d'un log en *Instance Graph* [Diamantini et al., 2016b] ; i.e., une représentation par graphe qui montre quelles activités sont exécutées séquentiellement ou en parallèle ; le *Behavioral PM* [Diamantini et al., 2016a] est une technique qui effectue un *graph clustering* pour extraire les sous-graphes fréquents. Cependant, les graphes ont un pouvoir d'expression limité aux séquences et aux parallélismes, et sont donc incapables de modéliser des situations de choix ou de boucles.

SUBDUE [Diamantini et al., 2013] est une technique un peu similaire qui, à partir d'un ensemble de modèles de processus transformés en graphes, extrait des comportements fréquents. Cependant, dans beaucoup de cas, soit il n'existe pas de modèles de processus *a priori*, soit ces modèles ne sont pas structurés.

De cette première catégorie de techniques, la LPM Discovery est à notre connaissance la seule qui génère des modèles ayant un pouvoir d'expression assez complet pour modéliser tous les comportements observables. De plus, cette technique utilise un event log comme donnée d'entrée. Par conséquent, de toutes les techniques citées qui génèrent des modèles "locaux", nous la considérons comme étant la technique la plus adaptée pour répondre à notre problématique. Dans la sous-section suivante, nous abordons les modèles déclaratifs et les comparons avec les modèles partiels construits par la technique de LPM Discovery.

5.2 Construction de modèles déclaratifs

Pour éviter la génération de modèles de processus "en spaghetti", il existe aussi les techniques qui construisent des modèles déclaratifs. Ces techniques de process discovery spécifient un ensemble de *règles* auxquelles les différentes exécutions du modèle doivent adhérer. Pour définir ces règles, elles utilisent un langage tel que Declare [Pesic and Van der

Aalst, 2006, Pesic et al., 2007, van der Aalst et al., 2009a], DPIL [Zeising et al., 2014] ou SCIFF [Alberti et al., 2008]. Les modèles déclaratifs se distinguent des modèles procéduraux dans le sens où ces derniers modélisent explicitement les comportements autorisés. De plus, ils sont très proches des modèles partiels construits par la technique de LPM Discovery.

Un exemple de contrainte est $Response(a, b)$, indiquant que si l'activité a est exécutée, alors l'activité b peut éventuellement être exécutée. Les approches existantes (Declare, DPIL et SCIFF) utilisent un ensemble prédéfini de contraintes, et extraient les relations entre activités pour lesquelles ces contraintes sont respectées, d'après un seuil de support. Cela signifie que les comportements qui intéressent un analyste doivent être spécifiées a priori, avant que la construction du modèle n'ait lieu. Cette particularité différencie ces techniques de celle de LPM discovery, dans laquelle des relations unaires sont étendues en relations binaires, puis ternaires, et puis itérativement en relations n-aires plus complexes tant que le seuil de support minimum est respecté.

Certaines variantes de DPIL et SCIFF autorisent la définition de contraintes complexes, qui spécifient des relations n-aires arbitraires entre activités ; les contraintes "standards" utilisées traditionnellement consistant en relations unaires et binaires, ne spécifiant pas de contraintes composées de plus de deux activités. Généralement, il faut combiner plusieurs contraintes pour spécifier une relation concernant plus de deux activités.

Le *Target-Branched Declare* (TBDeclare) [Di Ciccio et al., 2016], une variante de Declare, est une exception à cela dans la mesure où cette technique permet de définir des contraintes concernant plus de deux activités. Par exemple, $Response(a, \{b, c\})$ indique que si l'activité a est exécutée, l'activité b ou bien l'activité c peut éventuellement être exécutée.

$\mathcal{L}_1$
$\langle ..., a, ..., b, ..., c, ... \rangle^{40}$
$\langle ..., a, ..., b, ..., d, ... \rangle^{50}$

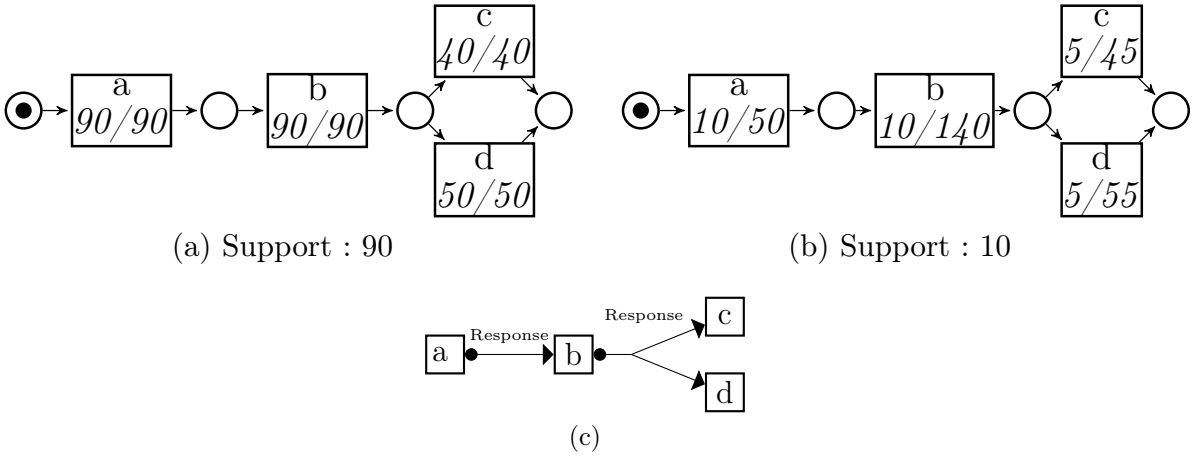
(a)

$\mathcal{L}_2$
$\langle ..., a, ..., b, ..., c, ... \rangle^{5}$
$\langle ..., a, ..., b, ..., d, ... \rangle^{5}$
$\langle ..., a, ..., b, ... \rangle^{40}$
$\langle ..., b, ..., c, ... \rangle^{40}$
$\langle ..., b, ..., d, ... \rangle^{50}$

(b)

FIGURE 5.1 – Exemples de log pour évaluer les différences entre combinaisons de contraintes Declare et un LPM.

Cependant, une combinaison de contraintes n'est pas équivalente à un LPM composé des mêmes activités en termes d'informations véhiculées. Le Figure 5.2 illustre cette différence. Considérons $\mathcal{L}_1$ et $\mathcal{L}_2$ les deux logs présentés dans la figure 5.1. $\langle a, b \rangle^x$ signifie que l'exécution des activités de a puis b s'est produite x fois dans le log. En utilisant un support minimum de 0,7, la règle $Response(a, b)$ serait extraite depuis $\mathcal{L}_1$ et $\mathcal{L}_2$, puisque dans chaque log toute exécution de l'activité a est éventuellement suivie de l'exécution de l'activité b. De plus, la règle $Response(b, \{c, d\})$ serait aussi extraite depuis les deux logs selon le même support minimum, puisque toutes les exécutions de l'activité b sont suivies soit de l'exécution de l'activité c soit de l'activité d dans $\mathcal{L}_1$ et 100 exécutions de l'activité b sur 140 sont suivies soit de l'exécution de l'activité c soit de l'activité d dans $\mathcal{L}_2$. Nous présentons la combinaison des contraintes $Response(a, b)$ et $Response(b, \{c, d\})$ dans la Figure 5.2c.



(a) Support : 90 (b) Support : 10

(c)

FIGURE 5.2 – *(a)* Un LPM construit à partir de $\mathcal{L}_1$, *(b)* un LPM construit à partir de $\mathcal{L}_2$ et *(c)* une combinaison de deux contraintes TBDeclare construites à partir de $\mathcal{L}_1$ et $\mathcal{L}_2$.

Contrairement à une combinaison de contraintes, le support d'un LPM est évalué pour l'ensemble du LPM. Prenons le LPM présenté dans la Figure 5.2b. Un LPM est modélisé par un réseau de Petri dont les transitions sont représentées sous la forme $(actX/Y)$, avec act le label de l'activité modélisée, Y le nombre d'events dans le log ayant exécuté act, et X le nombre d'events de Y rejouables dans le LPM. Par exemple dans le LPM de la Figure 5.2b, la première transition représente l'activité a. $Y = 50$ car il y a 50 exécutions de l'activité a dans $\mathcal{L}_2$, et $X = 10$ car sur les 50 exécutions totales, seulement 10 sont rejouables dans le LPM (les traces $\langle ..., a, ..., b, ..., c, ... \rangle$ et $\langle ..., a, ..., b, ..., d, ... \rangle$). Pour l'instant, considérons que le support d'un LPM est le nombre de fois que les exécutions possibles du LPM apparaissent dans le log. Les LPMs seront formellement définis dans la section suivante.

La méthode d'évaluation des LPMs est la raison du faible support de ce LPM. Construit à partir de $\mathcal{L}_2$, il indique que les exécutions de l'activité b qui suivent les exécutions de l'activité a ne sont pas les même exécutions qui précèdent les exécutions des activités c et d. Ce LPM contraste donc le LPM présenté dans la Figure 5.2a et construit à partir de $\mathcal{L}_1$, qui lui indique que les exécutions de l'activité b qui suivent les exécutions de l'activité a sont bien celles qui précédent les exécutions des activités c et d. Nous constatons donc qu'un LPM ne porte pas les mêmes informations que la combinaison de contraintes TBDeclare composée des mêmes activités.

Dans les travaux présentés dans ce manuscrit, nous nous intéressons à l'*utilité* des relations extraites. Une variante de la construction de modèles déclaratifs est la construction de modèles déclaratifs dit *"data-aware"* ; i.e., qui prennent en compte les différents attributs disponibles dans les events. MP-Declare [Burattin et al., 2016] étend Declare dans ce sens. Dans [Maggi et al., 2013], les auteurs proposent une technique pour construire un modèle MP-Declare depuis un event log. A partir de différents types de conditions, MP-Declare se concentre sur une analyse exploratoire, dont l'objectif est d'extraire des relations fortes entre le "control-flow" ; i.e., l'enchaînement des activités dans le processus, et les attributs. La différence avec la problématique à laquelle nous voulons répondre est que cette technique ne prend pas en compte un besoin métier. A l'inverse, nous voulons proposer une approche dont l'objectif est de donner la possibilité à un utilisateur d'interroger un log à la recherche de fragments de comportements qui répondent spécifiquement

à une question métier qui l'intéresse.

Nous venons de présenter les modèles locaux et déclaratifs pour modéliser les processus peu structurés. Parmi les techniques présentées, la *LPM Discovery* est celle qui possède les caractéristiques adéquates pour répondre à notre problématique. D'une part, elle est capable de prendre en charge des processus complexes par la modélisation de fragments de comportements, et ce dans une notation possédant une sémantique formelle. D'autre part, elle génère des modèles avec un fort pouvoir d'expression, capable de modéliser un grand nombre de constructions. Ce sont pour ces raisons que nous avons choisi de nous inspirer de la technique de LPM Discovery pour développer la seconde partie de nos travaux.

5.3 Local Process Model Discovery

Dans cette partie, nous présentons la technique de LPM Discovery, que nous utilisons dans la suite de ce manuscrit.

La technique de LPM Discovery utilise les réseaux de Petri et les process trees pour représenter les LPMs. Grâce à leur structure, les process trees sont utilisés pour la construction des LPMs, et le formalisme des réseaux de Petri pour l'évaluation et la représentation des LPMs.

Après avoir présenté les réseaux de Petri dans la sous-section 5.3.1, nous définissons dans la sous-section 5.3.2 les Local Process Models (LPMs), et présentons comment la technique de LPM Discovery génère un ensemble de LPMs.

Formalisation des Process Trees dans les LPMs

La technique de LPM Discovery, introduite dans [Tax et al., 2016b], génère des process trees dont la structure est légèrement différente de celle présentée par la Définition 13. D'une part, la procédure d'expansion utilisée a pour effet de ne construire que des process trees dont les nœuds-opérateurs ont exactement 2 enfants ; contre 2 ou plus dans la définition originale. Cela a pour effet de voir la contrainte stipulant qu'un nœud-opérateur de type Boucle doit avoir trois enfants (le do, le redo et la sortie) à n'avoir que le do et le redo comme enfants. L'activité de sortie est positionnée sur un nouveau nœud du process tree. Par conséquent, la condition selon laquelle un nœud-opérateur ne pouvait être enfant d'un nœud-opérateur du même type est aussi relâchée.
D'autre part, pour des raisons de complexité algorithmique lors de l'évaluation des LPMs, les auteurs ont volontairement exclu l'opérateur de choix inclusif OR ($\vee$) des opérateurs disponibles pour étendre un LPM. Cela restreint donc le pouvoir d'expression des LPMs. Cependant, la structure des LPMs et la modularité de la procédure d'expansion permettent très facilement d'inclure tout nouveau type d'opérateur.

5.3.1 Les réseaux de Petri

Pour rappel, les Réseau de Petri sont des modèles composés de *places* représentés par des cercles, de *transitions* représentés par des carrés (ou des rectangles) et d'arcs orientés liant places et transitions. Une transition peut représenter une activité et lorsqu'elle est déclenchée elle consomme un *jeton* (représenté par un point noir) dans chacune de ses

places d'entrée, et produit un jeton dans chacune de ses places de sortie. De cette façon, la distribution des jetons entre les différentes places représente les *états* du modèle, appelés *marquages*. Les activités silencieuses (τ) sont représentées par des rectangles noirs, et les marquages initial et final indiquent dans quel état le processus doit commencer et finir.

Nous définissons formellement un Réseau de Petri labellisé comme suit.

Définition 30 - Réseau de Petri labellisé :
Un Réseau de Petri labellisé $N{=}\langle P, T, F, \Sigma_M, l\rangle$ *est un tuple où* P *est un ensemble fini de places,* T *est un ensemble fini de transitions tel que* $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ *est un ensemble d'arcs orientés,* Σ_M *est un ensemble fini de labels représentant des activités, et* $l : T \nrightarrow \Sigma_M$ *est une fonction qui affecte un label à chaque transition. Il existe aussi des transitions non labellisées, appelées* τ*-transitions ou transitions silencieuses* $\tau \in T$ *avec* $\tau \notin dom(l)$.

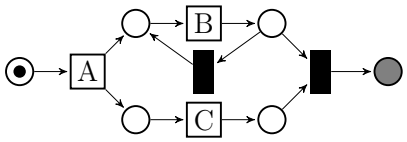
Pour un nœud $n \in P \cup T$, nous utilisons $\bullet n$ et $n\bullet$ pour désigner respectivement l'ensemble des nœuds d'entrée et de sortie de n, défini comme suit : $\bullet n{=}\{n' | (n', n) \in F\}$ et $n \bullet {=}\{n' | (n, n') \in F\}$.

L'état d'un Réseau de Petri est défini par son *marquage* $M \in \mathbb{N}^P$, qui est un multi-ensemble de places. Un marquage est graphiquement représenté par le fait de placer $M(p)$ jetons dans chaque place $p{\in}P$. Le réseau de Petri change d'état au cours des différents déclenchements de transitions. Une transition t est *franchissable* dans un marquage M si chaque place en entrée $p{\in} \bullet t$ contient au moins un jeton. Une fois qu'une transition est déclenchée, un jeton est retiré de chaque place en entrée de t et un jeton est ajouté dans chaque place en sortie de t, résultant dans un nouveau marquage $M'{=}M - \bullet t + t\bullet$. Le déclenchement d'une transition passant d'un marquage M à un marquage M' est noté $M \xrightarrow{t} M'$. $M_1 \xrightarrow{\sigma} M_2$ indique que le marquage M_2 peut être atteint depuis le marquage M_1 en déclenchant la séquence de transitions $\sigma{\in}T^*$.

Souvent, il est utile de considérer un réseau de Petri combiné à un marquage initial et un ensemble de marquages finaux. Cela permet de définir le langage du réseau de Petri et de vérifier si un comportement fait partie des comportements autorisés par le réseau de Petri.

Définition 31 - Réseau de Petri marqué avec marquages finaux :
Un Réseau de Petri marqué avec marquages finaux est un 3-uple $APN = (N, M_0, MF)$ *où* N *est un Réseau de Petri Labellisé,* $M_0 \in \mathbb{N}^P$ *représente le marquage initial de* N*, et* MF *est l'ensemble des marquages finaux possibles, tels que* $\forall M_1, M_2 \in MF, M_1 \nsubseteq M_2$. *Nous appelons une séquence* $\sigma \in T^*$ *une trace du réseau de Petri marqué avec marquages finaux* APN *si* $M_0 \xrightarrow{\sigma} M_f$ *pour un marquage final* $M_f \in MF$. *Le langage d'un réseau de Petri marqué avec marquages finaux, noté* $\mathfrak{L}(APN)$*, est l'ensemble de toutes les séquences de labels appartenant à ses traces ; i.e., si* σ *est un trace de APN, alors* $l(\sigma) \in \mathfrak{L}(APN)$.

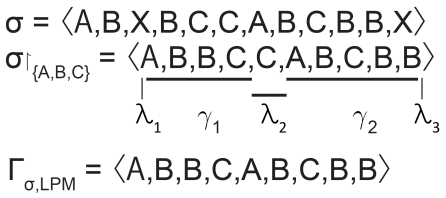
Nous présentons dans la Figure 5.3a un exemple de réseau de Petri marqué avec marquages finaux. Les places qui appartiennent au marquage initial contiennent un jeton chacune, et les places qui appartiennent au marquage final ont un label f_i avec i comme identifiant, ou sont simplement représentée par le symbole ● dans le cas où il n'existe qu'un seul marquage final. Le langage de ce réseau de Petri marqué avec marquages finaux avec $\Sigma_M{=}\{A, B, C\}$, est $\{\langle A, B, C\rangle, \langle A, C, B\rangle \langle A, B, B, C\rangle, \langle A, B, C, B\rangle, \langle A, B, B, B, C\rangle, ...\}$. Nous définissons le *langage n-limité* d'un LPM, noté $\mathfrak{L}_n(LPM)$, comme étant le langage du LPM composé des traces de taille égale ou inférieure à n.



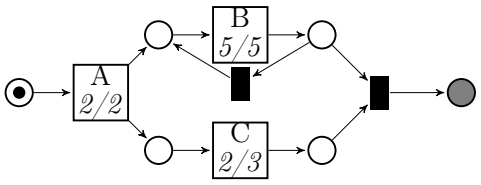
(a)

event id	activity	time	cost
1	A	26-3-2017 13 :00	100€
2	B	26-3-2017 13 :25	500€
3	X	26-3-2017 13 :27	60€
4	B	26-3-2017 13 :30	400€
5	C	26-3-2017 13 :35	100€
6	C	26-3-2017 13 :42	500€
7	A	26-3-2017 15 :26	300€
8	B	26-3-2017 15 :27	50€
9	C	26-3-2017 15 :47	100€
10	B	26-3-2017 16 :10	250€
11	B	26-3-2017 16 :52	300€
12	X	26-3-2017 16 :59	10€

(b)



(c)



(d)

FIGURE 5.3 – *(a)* Exemple de réseau de Petri marqué avec marquages finaux APN_1. *(b)* Trace σ issue d'un event log $\mathcal{L}$. *(c)* Segmentation de σ sur APN_1. *(d)* LPM LPM obtenu en évaluant APN_1 sur σ.

5.3.2 Local Process Models

En nous inspirant de [Tax et al., 2016b], nous définissons un Local Process Model comme suit. Un *Local Process Model*, noté LPM, décrit un fragment de comportement fréquent dans un event log, comportant généralement entre 3 et 5 activités. Au lieu de décrire tout le comportement observé dans les différentes traces, un LPM a pour objectif de représenter une partie de ces traces qui se répète souvent. Un LPM LN représente un comportement défini sur l'ensemble des labels Σ_M et possède le langage $\mathfrak{L}(LN)$.

La technique de LPM Discovery propose une approche itérative pour construire des LPMs, et consiste en 4 étapes : (1) la *génération*, (2) l'*évaluation*, (3) la *sélection* et (4) l'*expansion*. Nous représentons ces étapes dans la Figure 5.4.

1. **La génération** :
L'objectif de la première étape est de générer l'ensemble des LPMs candidats CM_1. Chaque candidat est un process tree initial composé d'un nœud-feuille représentant une activité a appartenant à l'ensemble des activités du log, $a \in \Sigma_{\mathcal{L}}$. Un LPM composé d'une seule activité est un LPM *"initial"*. Il y a autant de LPMs initiaux candidats que d'activités dans $\Sigma_{\mathcal{L}}$.

2. **L'évaluation** :
Dans l'étape d'évaluation, les différents LPM sont évalués selon une liste de mesures. Avant de présenter la liste de mesures utilisées par la technique de LPM Discovery, nous détaillons la méthode d'évaluation d'un LPM.

Pour évaluer un LPM sur un log $\mathcal{L}$, chaque trace $\sigma \in \mathcal{L}$ est d'abord projetée sur l'ensemble d'activités Σ_M du LPM ; i.e., $\sigma'{=}\sigma \restriction_{\Sigma_M}$.

Chaque trace projetée est ensuite *segmentée* en γ-segments, les segments de la trace qui sont rejouables dans le LPM, et en λ-segments, les segments de la trace qui ne sont

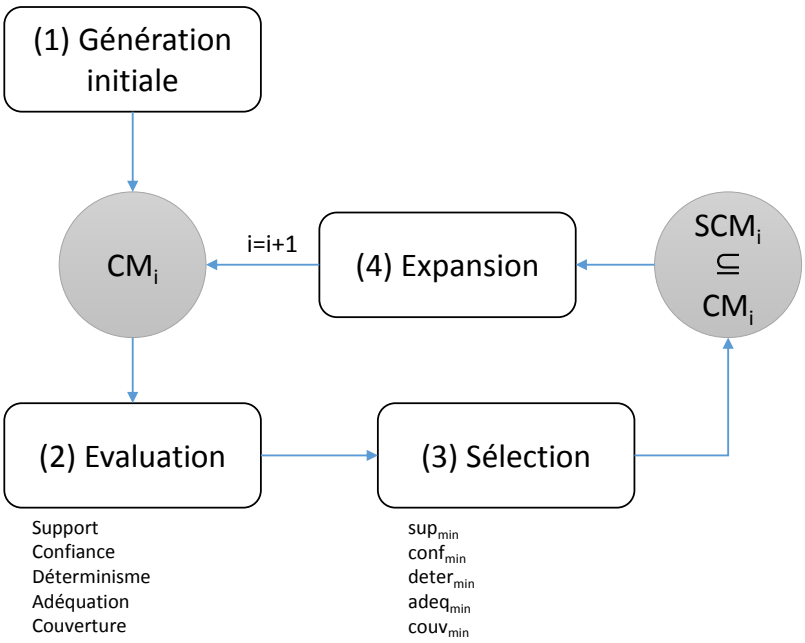


FIGURE 5.4 – Fonctionnement de la technique de LPM Discovery.

pas rejouables dans le LPM ; i.e., $\sigma'{=}\lambda_1\gamma_1\lambda_2\gamma_2\lambda_n\gamma_n\lambda_{n+1}$ tel que $\gamma_i \in \mathfrak{L}(LPM)$ et $\lambda_i \notin \mathfrak{L}(LPM)$. Nous définissons $\Gamma_{\sigma,LPM}$ la fonction qui projette la trace σ sur les activités du LPM et obtient les segments qui sont rejouables dans le LPM ; i.e., $\Gamma_{\sigma,LPM}{=}\gamma_1 \cdot \gamma_2 \cdots \gamma_n$. Considérons le LPM présenté dans la Figure 5.3a, et la trace présentée dans la Figure 5.3b, avec $\pi_{activity}(\sigma){=}\langle A, B, X, B, C, C, A, B, C, B, B, X\rangle$. La projection de σ sur les activités du LPM donne $\pi_{activity}(\sigma)\restriction_{\{A,B,C\}}{=}\langle A, B, B, C, C, A, B, C, B, B\rangle$. Nous présentons dans la Figure 5.3c la segmentation de la projection de la trace sur le LPM, telle que $\pi_{activity}(\Gamma_{\sigma,LPM}){=}\langle A, B, B, C, A, B, C, B, B\rangle$. La segmentation commence avec un λ-segment vide, suivi d'un γ-segment $\gamma_1{=}\langle A, B, B, C\rangle$, qui représente une exécution du LPM depuis le marquage initial jusqu'au marquage final. La seconde execution de l'activité C dans σ ne peut pas être rejouée dans le LPM, résultant en un λ-segment $\lambda_2{=}\langle C\rangle$. $\gamma_2{=}\langle A, B, C, B, B\rangle$ est un nouveau γ-segment car il représente une autre exécution du LPM. La segmentation se termine avec un dernier λ-segment vide $\lambda_3{=}\langle\rangle$. Nous adaptons la fonction de segmentation à un event log, tel que $\Gamma_{\mathcal{L},LPM}{=}\{\Gamma_{\sigma,LPM} | \sigma \in \mathcal{L}\}$.
Nous présentons dans la Figure 5.3d le LPM obtenu après évaluation du réseau de Petri marqué avec marquages finaux APN_1 sur la trace $\sigma \in \mathcal{L}$. Il indique que 2 occurrences de l'activité A sur 2, 5 occurrences de l'activité B sur 5, et 2 occurrences de l'activité C sur 3 sont rejouables dans APN_1.

Nous définissons $activities(a, \mathcal{L}){=}|\{e{\in}\sigma | (\sigma \in \mathcal{L}){\wedge}(\pi_{activities}(e){=}a)\}|$ la fonction qui affecte à chaque activité de l'event log son nombre d'occurrences. $activities(\mathcal{L})$ représente le multi-ensemble composé du nombre d'occurrences de toutes les activités qui apparaissent dans $\mathcal{L}$. $events(L){=}\{e{\in}\sigma | \sigma \in \mathcal{L}\}$ est l'ensemble des events de $\mathcal{L}$. Il est à noter que les fonctions *activities* et *events* peuvent aussi être appliquées à $\Gamma_{\mathcal{L},LPM}$, qui est en soi un event log.

La technique LPM Discovery évalue les différents LPMs selon 5 mesures : le *support*, la *confiance*, l'*adéquation*, *le déterminisme* et la *couverture*. Nous définissons $\Gamma^{\gamma}_{\sigma,LPM}$ le multi-ensemble composé des γ-segments de la trace σ rejouables dans le LPM, et $\Gamma^{k}_{\sigma,LPM}$ la fonc-

tion qui donne le nombre de γ-segments dans $\Gamma^{\gamma}_{\sigma,LPM}$. $\#_a(s)$ est une fonction qui renvoie le nombre d'occurrences de l'élément a dans la séquence s ; e.g., pour $s = \langle a, b, c, a, b\rangle$, $\#_a(s)$=2. Nous l'élevons aux multi-ensembles $\#_a(\mathcal{L})$; e.g, pour $\mathcal{L}$=$[\langle a, b, c\rangle^2, \langle b, a, c\rangle^3]$, $\#_a(\mathcal{L})$=5. Dans la suite, nous illustrons les différentes mesures avec un exemple composé du LPM présenté dans la Figure 5.3a, que nous notons LN, et $\mathcal{L}$ le log constitué de la trace présentée dans la Figure 5.3b.

—**Le support** : représente le rapport du nombre de fragments du log qui peuvent être rejoués dans le LPM sur ce même nombre plus un. Plus le LPM représente des fragments observés, plus il a de chance d'être intéressant. La valeur du support d'un LPM est définie sur un intervalle [0 ;1) comme suit :

$$support(LN, \mathcal{L}) = \frac{\sum\limits_{\sigma\in\mathcal{L}} \Gamma^{k}_{\sigma,LPM}}{(\sum\limits_{\sigma\in\mathcal{L}} \Gamma^{k}_{\sigma,LPM}) + 1} \tag{5.1}$$

Dans l'exemple, $support(LN, \mathcal{L}) = \frac{2}{3} \approx 0,66$.

—**La confiance** : La confiance d'une activité $a \in \Sigma_M$ dans un LPM LN pour un log $\mathcal{L}$ est le ratio d'events composés de l'activité a rejouables dans LN, formalisé comme suit :

$$confiance(a, \mathcal{L}) = \frac{\sum\limits_{\sigma\in\mathcal{L}} \#_a(\Gamma^{\gamma}_{\sigma,LPM})}{\#_a(\mathcal{L})} \tag{5.2}$$

Dans l'exemple, $confiance(c, \mathcal{L}) = \frac{2}{3} \approx 0,66$.

Pour être plus sensible aux valeurs plus petites que la moyenne, la confiance d'un LPN LN dans un log $\mathcal{L}$ est définie comme la moyenne harmonique des confiances de chaque activité de Σ_M.

$$confiance(LN, \mathcal{L}) = \frac{|\Sigma_M|}{\sum\limits_{a\in\Sigma_M} \frac{1}{confiance(a,LN)}} \tag{5.3}$$

Dans l'exemple, $confiance(LN, \mathcal{L}) = \frac{3}{\frac{1}{1}+\frac{1}{1}+\frac{1}{0,66}} \approx 0,85$.

—**L'adéquation** : représente le rapport des comportements observés dans le log par rapport aux comportements autorisés par le LPM. Un LPM contenant une boucle pouvant autoriser un nombre infini de comportements, l'adéquation d'un LPM à un log est défini par rapport au langage n-limité du LPM, comme suit :

$$adéquation_n(LN, \mathcal{L}) = \frac{|\{s \in \mathfrak{L}_n(LN) | \exists\sigma\in\mathcal{L} : s \in \Gamma^{\gamma}_{\sigma,LPM}\}|}{|\mathfrak{L}_n(LN)|} \tag{5.4}$$

Dans l'exemple, $adéquation_5(LN, \mathcal{L}) = \frac{2}{9} \approx 0,22$.

—**Le déterminisme** : représente la capacité prédictive d'un LPM à partir d'un marquage de ce LPM. Par exemple, si le langage d'un LPM comporte les séquences $\langle a, b, c, d\rangle$ et $\langle g, b, e, f\rangle$; la continué d'une trace ayant exécuté l'activité b peut être soit l'exécution de l'activité c ou de l'activité e, laissant un degré d'incertitude, ce

qui n'est pas voulu. Soit $\mathcal{R}(LN)$ l'ensemble des marquages atteignables du LPM LN, $\mathcal{W}_{\mathcal{L}} : \mathcal{R}(LN) \to \mathbb{N}$ est la fonction qui assigne à chaque marquage atteignable le nombre de fois où il a été atteint en rejouant les events de $\Gamma_{\mathcal{L},LPM}$. $\mathcal{D} : \mathcal{R}(LN)$ est la fonction qui assigne le nombre de transitions franchissables pour un marquage donné. Le déterminisme d'un LPM est donc défini comme suit :

$$déterminisme(LN, \mathcal{L}) = \frac{\sum_{m \in \mathcal{R}(LN)} \mathcal{W}_{\mathcal{L}}(m)}{\sum_{m \in \mathcal{R}(LN)} \mathcal{W}_{\mathcal{L}}(m).\mathcal{D}(m)} \tag{5.5}$$

Dans l'exemple, $déterminisme(LN, \mathcal{L}) = \frac{13}{16} \approx 0,81$.

—**La couverture** : représente le ratio du nombre d'events contenus dans la projection du log $\mathcal{L}$ sur Σ_M sur le nombre total d'events dans le log. Pour $\#_*(\mathcal{L})$ le nombre total d'events dans le log $\mathcal{L}$, la couverture d'un LPM LN est définie comme suit :

$$couverture(LN, \mathcal{L}) = \frac{\#_*(\mathcal{L} \restriction_{\Sigma_M})}{\#_*(\mathcal{L})} \tag{5.6}$$

Dans l'exemple, $couverture(LN, \mathcal{L}) = \frac{10}{12} \approx 0,83$.

3. **La sélection** :
A partir des mesures définies, un ensemble de LPMs candidats $SCM_i \subseteq CM_i$ est sélectionné. Cet ensemble est ajouté à l'ensemble SM représentant les LPMs extraits, $SM = SM \cup SCM_i$.
C'est à partir de là que le processus devient itératif. L'étape suivante consiste à *étendre* l'ensemble SCM_i via une procédure d'expansion. Dès lors que l'ensemble de LPMs candidats devient vide ; i.e., $SCM_i = \emptyset$, ou que la technique a itéré assez de fois ; i.e., $i \geq max\_iterations$, la technique s'arrête et extrait l'ensemble des LPMs contenus dans SM. Généralement le nombre d'itérations maximum est fixé à 4 ou 5, pour que les temps d'exécution restent acceptables et que les modèles générés restent "partiels".

4. **L'expansion** :
La procédure d'expansion consiste à étendre un ensemble de LPMs candidats SCM_i en un ensemble CM_{i+1} plus grand.
Pour un process tree, une expansion est le fait de remplacer un nœud-feuille représentant l'activité a par un nœud-opérateur avec deux enfants : l'un est le nœud-feuille remplacé, et l'autre un nouveau nœud-feuille représentant une activité b du log ; $b \in \Sigma_L$, avec $a \neq b$.

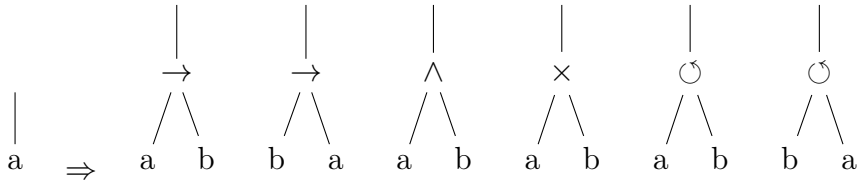


FIGURE 5.5 – Ensemble des expansions possibles du noeud-feuille a avec l'activité b [Tax et al., 2016b].

La Figure 5.5 présente l'ensemble des expansions possibles depuis un LPM. A partir de l'ensemble d'opérateurs $\{\rightarrow, \wedge, \times, \circlearrowleft\}$, 6 expansions sont possibles pour étendre un noeud-feuille représentant l'activité a avec une nouvelle activité b. Les opérateurs de type AND

et XOR sont des opérateurs *symétriques* ; i.e., la position d'insertion de la nouvelle activité b n'a pas d'impact sur le langage du nouveau LPM. Par contre, les opérateurs de type Séquence et Boucle n'étant pas symétriques, il y a deux expansions possibles pour chacun de ces opérateurs. Des contraintes sur cette procédure d'expansion ont été développées pour éviter de générer deux fois le même LPM au cours des différentes expansions.

Pour un log $\mathcal{L}$, nous définissons $LPMS(\mathcal{L})$ l'espace de recherche de $\mathcal{L}$; i.e., l'ensemble des LPMs qu'il est possible de construire à partir des activités de $\mathcal{L}$. Cet espace de recherche croît avec le nombre d'activités dans le log, $|\Sigma_L|$ et le nombre de nœuds-feuille autorisés dans les LPMs. Pour un LPM donné, il est possible de générer $|\Sigma_M| \times 6 \times (|\Sigma_L| - |\Sigma_M|)$ nouveaux LPMs. Pour répondre à cette problématique, la technique de LPM Discovery se base sur la propriété anti-monotone du support [Tax et al., 2016b] et sur certaines approches heuristiques [Tax et al., 2016d] pour élaguer l'espace de recherche et réduire les temps de calculs. Pour un LPM LN et une condition c, nous définissons que c est anti-monotone si le fait que LN ne respecte pas c implique que toute expansion de LN ne respecte pas c non plus.

Une fois l'expansion effectuée pour chaque LPM de SCM_i, la technique retourne à l'étape d'évaluation.

Ces quatre étapes ont pour objectif d'extraire l'ensemble des LPM fréquents. Le problème d'extraction de modèles de processus partiels est donc défini comme suit :

Définition 32 - Problème d'Extraction de Modèles de Processus Partiels :
Soit sup_{min}, $conf_{min}$, $adeq_{min}$, $deter_{min}$ et $couv_{min}$ des seuils minimum pour chacune des mesures présentées et score la valeur pondérée des mesures utilisées pour évaluer un LPM. Un LPM est intéressant dès lors qu'il fournit des informations pertinentes pour l'analyste qui l'extrait ; i.e., si chacune des mesures qui l'évalue a une valeur supérieure ou égale à son seuil correspondant prédéfini ou s'il fait partie des top−k LPMs avec le plus gros score. Le problème d'extraction de modèles de processus partiels est la tâche qui consiste à extraire la collection des LPMs intéressants à partir d'un log $\mathcal{L}$.

Conclusion

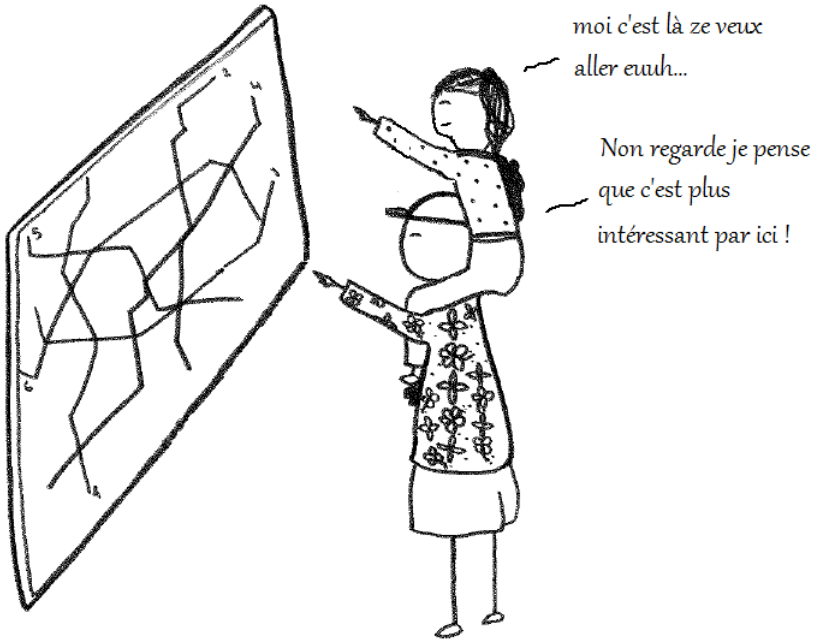
Dans ce chapitre, nous avons tout d'abord présenté les modèles locaux et déclaratifs, deux alternatives aux modèles globaux, habituellement générés par les techniques de process mining, pour modéliser les processus peu structurés. Parmi les techniques présentées, nous avons retenu la *LPM Discovery* car elle est capable de prendre en charge des processus complexes par la modélisation de fragments de comportements, et ce dans une notation possédant une sémantique formelle. En plus de cela, elle génère des modèles avec un fort pouvoir d'expression, capable de modéliser un grand nombre de comportements.

Nous avons ensuite vu en détail la technique de LPM Discovery, en présentant les réseaux de Petri et la procédure utilisée pour construire les LPMs.

Chapitre 6

Extraction de Modèles de Processus Partiels Profitables

Sommaire



Introduction

Dans la première partie de ce manuscrit, nous avons proposé une approche pour détecter des dépendances intéressantes entre les activités d'un processus. Après avoir mené différentes expérimentations, nous avons conclu que notre approche était moins efficace si elle était utilisée sur un modèle peu structuré. Pour palier ce problème, nous décidons d'utiliser d'autres alternatives aux modèle "standard" qui modélisent les processus de bout en bout. Dans le chapitre précédent, nous avons vu que les modèles "locaux" et les modèles "déclaratifs" permettaient de gérer les processus peu structurés. Parmi les techniques disponibles, nous avons décidé de nous inspirer de la *LPM Discovery*, une technique qui modélise des processus partiels ; i.e., des fragments de processus fréquents. Toujours dans l'optique de modéliser des comportements intéressants d'un point de vue métier, l'objectif de ce chapitre est de présenter une approche pour extraire des fragments de processus profitables.

Ce chapitre est organisé comme suit. Dans la section 6.1, nous définissons le problème d'extraction de modèles de processus partiels profitables. La proposition de ce nouveau problème est motivé par le besoin d'extraire des modèles "locaux" dont l'intérêt dépend de l'analyste qui génère le modèle. Pour cela, nous proposons la *High-Utility Local Process Model Discovery* (HU-LPM Discovery), une approche dans laquelle nous définissons un ensemble de contraintes et fonctions de profit pour spécifier par quelles parties du processus un utilisateur est intéressé.

Enfin, dans la section 6.2, nous proposons la *Heuristic High-Utility Local Process Model Discovery* (Heuristic HU-LPM Discovery), une technique qui utilise un ensemble d'heuristiques pour l'extraction de modèles de processus partiels profitables. Selon l'event log utilisé, il arrive que l'extraction nécessite des temps de calcul importants. Par conséquent, l'utilisation de méthodes approchées est une alternative que nous explorons.

6.1 Définition du problème d'extraction de modèles de processus partiels profitables

Dans cette section, nous envisageons l'extraction de Local Process Models (LPMs) d'un point de vue "métier", en fonction des besoins de l'analyste qui effectue l'extraction. Pour cela, nous proposons une nouvelle technique d'extraction, la HU-LPM Discovery, qui utilise une combinaison de *contraintes* et de *fonctions* de profit. Ces contraintes et fonctions de profit ont la caractéristique de pouvoir être définies sur différents *périmètres* ; i.e. des niveaux d'abstraction utilisés en fonction de l'analyse à effectuer. Alors que l'extraction classique de LPMs choisit et classe les LPMs en fonction de leur support respectif, le profit permet une sélection et un classement basés sur l'intérêt propre à chaque analyste qui interroge un même event log.

Dans un premier temps, nous définissons dans la section 6.1.1 le domaine d'application sur lequel les différentes contraintes et fonctions de profit sont définies. Dans un second temps, nous introduisons les différentes contraintes et fonctions de profit selon chaque périmètre du domaine d'application.

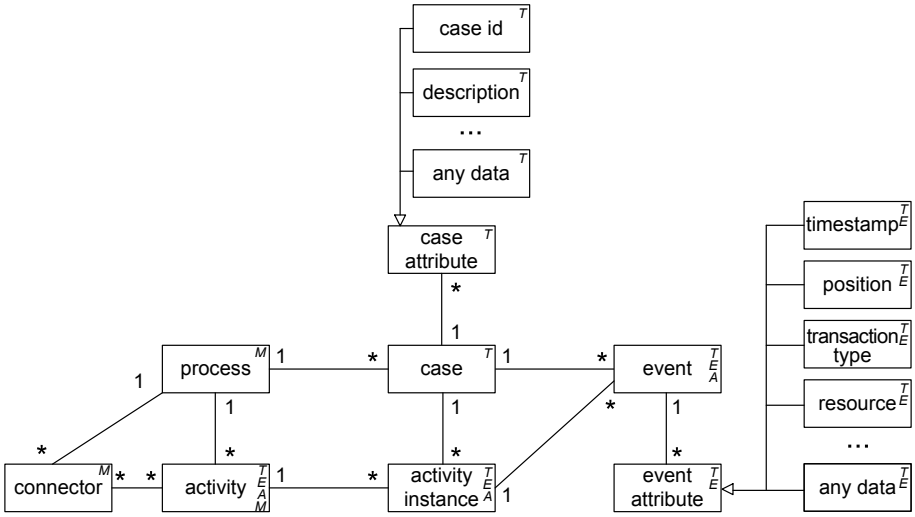


FIGURE 6.1 – Concepts relatifs à la composition d'un event log modélisés par un diagramme de classe [van der Aalst, 2016] et annotés par le(s) périmètre(s) dans le(s)quel(s) ils sont contenus.

6.1.1 Domaine d'application

Dans nos travaux, nous définissons le domaine $T.E.A.M$, composé de 4 périmètres d'application sur lesquels les contraintes et fonctions de profit peuvent être définies : le périmètre *trace* (T), le périmètre *event* (E), le périmètre *activité* (A) et le périmètre *modèle* (M). Dans la Figure 6.1 nous présentons un diagramme de classe représentant le domaine $T.E.A.M$. Une trace (*case*) est le résultat de l'exécution d'un processus (*process*). Elle possède plusieurs attributs (*case attribute*) et est composée d'events (*event*). Ces derniers représentent l'exécution d'une activité (*activity instance*) et possèdent une liste d'attributs (*event attribute*). Le processus modélisé est composé d'activités (*activity*) reliées les unes aux autres par des opérateurs (*connectors*). Chacune des classes est annotée par le périmètre qui peut y accéder ; e.g., le label d'une activité est accessible par tous les périmètres du domaine $T.E.A.M$ alors que le timestamp d'un event est accessible seulement depuis les périmètres trace et event. Les différents niveaux d'accessibilité sont détaillés dans la sous-section suivante.

6.1.2 Contraintes et fonctions de profit

Pour définir le caractère "intéressant" des LPMs d'un point de vue métier, nous utilisons des contraintes et fonctions de profit. Les contraintes de profit peuvent par exemple être utilisées pour extraire des fragments de comportements qui mènent au refus d'une demande de prêt bancaire, ou pour extraire des fragments de comportements qui décrivent seulement des demandes de prêt dont le montant est supérieur à 15 000 euros et seulement ceux-là. Les contraintes de profit sont des conditions que les LPMs doivent satisfaire. Par conséquent, nous définissons une contrainte comme étant une fonction dont le résultat est 1 lorsque la condition est satisfaite, 0 sinon. Plus formellement, une contrainte est définie comme une fonction $c : X \rightarrow \{0, 1\}$, où X est le *périmètre* sur lequel la fonction est appliquée.

Les fonctions de profit indiquent dans quelle mesure un LPM est supposé être intéressant du point de vue de l'analyste qui l'extrait. Elles peuvent être utilisées pour

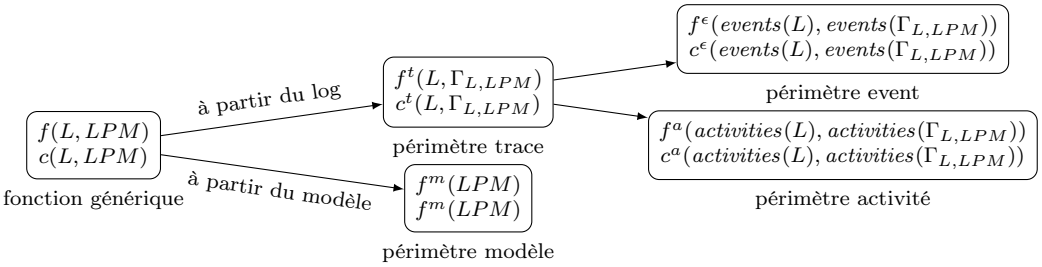


FIGURE 6.2 – Taxonomie des contraintes et fonctions de profit sur les différents périmètres.

extraire des fragments de comportements responsables de coûts financiers importants ou de retards de paiement. Elles sont définies comme des fonctions $f : X \to \mathbb{R}$. Tout comme les contraintes de profit, les fonctions de profit peuvent être définies sur les 4 périmètres d'application que nous avons définis dans la sous-section 6.1.1. La différence essentielle qu'il existe entre les contraintes et les fonctions de profit est que, là où les contraintes opèrent comme des *conditions nécessaires* ; i.e., les LPMs qui ne les respectent pas sont par défaut considérés comme inintéressants, les fonctions de profits opèrent plus comme des *préférences*.

De manière générale, les contraintes et fonctions de profit opérant sur les périmètres trace, event et activité sont définies sur une combinaison d'un LPM et d'un event log. Elles évaluent le profit d'un LPM à partir d'un log $\mathcal{L}$ et des fragments rejouables dans le LPM, donnés par la fonction de segmentation $\Gamma_{\mathcal{L},LPM}$. Les différents périmètres sur lesquels elles peuvent être définies se distinguent par la partie du log $\mathcal{L}$ et de la segmentation $\Gamma_{\mathcal{L},LPM}$ sur lesquelles elles vont opérer. Les fonctions et contraintes de profit définies sur le périmètre modèle ne considèrent que le LPM, et ignore le log.

Quelle(s) contrainte(s) ou fonction(s), et quel(s) périmètre(s) utiliser dépend donc de la réponse à laquelle l'analyste souhaite répondre. Dans la suite de cette section, nous formalisons les contraintes et fonction de profit en fonction des 4 périmètres d'application. Nous illustrons chacune de ces définitions en utilisant l'exemple suivant. Un analyste souhaite analyser un processus hospitalier. Il est particulièrement intéressé par les étapes du processus dans lesquelles beaucoup d'heures-homme sont consommées [1]. Il considère qu'une étape est un ensemble d'activités exécutées dans une fenêtre de temps prédéfinie. L'analyste a à sa disposition un event log composé des différentes exécutions du processus observées dans son hôpital, dans lesquelles la consommation d'heures-homme est enregistrée pour chaque event. Une analyse sur laquelle il souhaite se concentrer porte sur les activités ou séquences d'activités qui sont exécutées plus d'une fois dans une même trace. Par conséquent, il va essayer de mettre en évidence les comportements composés de boucles. Enfin, certaines activités observées sont des activités exécutées automatiquement par le système d'information de l'hôpital ; e.g., des activités de validation ou de contrôle. Puisqu'aucune heure-homme n'est consommée par ces activités, l'analyste ne souhaite pas les inclure dans l'analyse.

Nous présentons dans la Figure 6.2 une taxonomie des contraintes et fonctions de profit sur les différents périmètres du domaine d'application ; et les détaillons dans les sous-sections suivantes.

1. Unité de mesure correspondant au travail d'une personne pendant une heure.

6.1.2.1 Contraintes et fonctions de profit définies sur le périmètre trace

Les fonctions de profit définies sur le périmètre trace sont les fonctions de profit les plus générales définies sur un event log. Ces fonctions évaluent le profit d'un LPM en agrégeant le profit des fragments de traces qui sont rejouables dans le LPM. Le profit de ces segments peut dépendre des events contenus dans ces fragments, de n'importe quel attribut de ces events voire même de n'importe quel attribut de la trace à partir de laquelle le fragment est extrait.

Par exemple, une fonction de profit définie sur le périmètre trace peut être utilisée pour construire les LPMs décrivant des fragments de comportements responsables d'une grande partie des dépenses financières de la trace, ou dans lesquels la majorité du temps d'exécution de la trace est concentrée.

Une fonction de profit définie sur le périmètre trace est une fonction $f^t(\mathcal{L}, \Gamma_{\mathcal{L},LPM})$ indiquant le profit des différents fragments de trace dans $\Gamma_{\mathcal{L},LPM}$. Considérons une trace σ possédant un attribut $total\_cost$ qui indique le coût financier total généré par la trace, et composée d'events possédant un attribut *cost* représentant leur coût. Un exemple de fonction de profit définie sur le périmètre trace est $f_1^t(\mathcal{L}, \Gamma_{\mathcal{L},LPM}) = \sum_{\sigma' \in \Gamma_{\mathcal{L},LPM}} \sum_{e \in \sigma'} \frac{\pi_{cost}(e)}{\phi_{total\_cost}(\sigma')}$, dont l'objectif est de permettre l'extraction de LPMs responsables d'une majeure partie des dépenses financières de la trace. Un autre exemple de fonction est $f_2^t(\mathcal{L}, \Gamma_{\mathcal{L},LPM}) = \sum_{\sigma' \in \Gamma_{\mathcal{L},LPM}} \pi_{time}(\sigma'(|\sigma'|)) - \pi_{time}(\sigma'(1))$ qui extrait les LPMs dont l'exécution des comportements modélisés prend généralement beaucoup de temps.

Les contraintes de profit définies sur le périmètre trace imposent des conditions sur $\Gamma_{\mathcal{L},LPM}$. Toutes les fonctions de profit peuvent être transformées en contraintes de profit en ajoutant des seuils minimaux ou maximaux sur la valeur du résultat.

Sur l'exemple du processus hospitalier, la condition définie par l'analyste exigeant que les fragments de comportements modélisés par les LPMs représentent un ensemble d'activités exécutées dans une fenêtre de temps prédéfinie, peut être définie à partir de la contrainte de profit suivante : $c^t_{exemple}(\mathcal{L}, LPM) = \frac{f_2^t(\mathcal{L}, \Gamma(\mathcal{L},LPM))}{|\Gamma_{\mathcal{L},LPM}|} \leq max\_time$; avec max_time la durée de la fenêtre de temps autorisée.

6.1.2.2 Contraintes et fonctions de profit définies sur le périmètre event

Les contraintes et fonctions de profits définies sur le périmètre event peuvent être utilisées lorsque le besoin métier exprimé par une partie prenante du processus ne concerne que les attributs des events, et ne concerne pas le contexte de la trace dans laquelle ces events sont exécutés. Ces fonctions peuvent être utilisées par exemple pour extraire des LPMs décrivant des fragments de comportements avec un coût financier élevé. Cette fonction diffère de celle présentée dans le périmètre trace dans le sens où celle-ci ne compare pas le coût financier du segment rejouable par rapport au coût financier global de la trace de laquelle il est extrait. Les contraintes de profit définies sur ce périmètre peuvent être utilisées par exemple pour n'extraire que les LPMs composés d'activités exécutées par une certaine ressource ou à certains moments de la journée. Le périmètre trace est plus expressif que le périmètre event, permettant de définir des contraintes et fonctions de profit plus complexes ; mais en même temps reste plus compliqué à utiliser.

Une fonction de profit définie sur le périmètre event est une fonction $f^e(events(\mathcal{L}), events(\Gamma_{\mathcal{L},LPM}))$ indiquant le profit des différents events compris dans $\Gamma_{\mathcal{L},LPM}$. Considérons une trace σ composée d'events possédant un attribut *cost* représentant leur coût. Un exemple de fonction de profit définie sur le périmètre

event est $f_1^e(events(\mathcal{L}), events(\Gamma_{\mathcal{L},LPM})) = \sum_{e \in events(\Gamma_{\mathcal{L},LPM})} \pi_{cost}(e)$, qui permet d'extraire des LPMs avec un coût financier important. Il est à noter que le log $\mathcal{L}$ est compris dans le domaine de définition, permettant par exemple de formuler une fonction de profit optimisant la part de profit généré par activité $f_2^e(events(\mathcal{L}), events(\Gamma_{\mathcal{L},LPM})) = \sum_{a \in \Sigma_{\mathcal{L}}} \frac{\sum_{e \in events(\Gamma_{\mathcal{L},LPM})} \pi_{cost}(e) \times (\pi_{activity}(e)=a)}{\sum_{e \in events(\mathcal{L})} \pi_{cost}(e) \times (\pi_{activity}(e)=a)}$.

Dans l'exemple du processus hospitalier, la volonté de l'analyste d'extraire des LPMs dont le profit est représenté par le nombre d'heures-homme consommées peut être représentée par une fonction de profit définie sur le périmètre event $f_{exemple}^e(events(\mathcal{L}), events(\Gamma_{\mathcal{L},LPM})) = \sum_{e \in events(\Gamma_{\mathcal{L},LPM})} \pi_{man\_hours}(e)$; avec man_hours l'attribut de l'event représentant le nombre d'heure-homme consommée par la ressource pour exécuter l'activité.

Une contrainte de profit définie sur le périmètre event est une fonction $c^e(events(\mathcal{L}), events(\Gamma_{\mathcal{L},LPM}))$ et exprime des conditions sur les events dans $events(\Gamma_{\mathcal{L},LPM})$. Un exemple de contrainte est $c_1^e(events(\mathcal{L}), events(\Gamma_{\mathcal{L},LPM})) = f_1^e(events(\mathcal{L}), events(\Gamma_{\mathcal{L},LPM})) \geq 500$, qui indique que les LPMs extraits doivent avoir un coût d'au moins 500 €, ou $c_2^e(events(\mathcal{L}), events(\Gamma_{\mathcal{L},LPM})) = \forall e \in \Gamma_{\mathcal{L},LPM}, \pi_{cost}(e) \geq 100$, qui indique que les LPMs extraits ne doivent jamais rejouer des events dont le coût est inférieur à 100 €. Il est à noter que c_2^e n'est pas respectée par le LPM et la trace présentés dans les Figures 5.3a et 5.3b, où l'event avec l'Id 8 est rejouable dans le LPM mais possède un coût de seulement 50 €. Par conséquent, ce LPM ne pourrait être extrait si la contrainte c_2^e est exprimée.

6.1.2.3 Contraintes et fonctions de profit définies sur le périmètre activité

Les contraintes et fonctions de profit qui opèrent sur le périmètre activité sont définies par rapport au nombre d'occurrences de chaque activité dans le log $\mathcal{L}$ et dans la segmentation $\Gamma_{\mathcal{L},LPM}$. Les fonctions de profit définies sur ce périmètre peuvent par exemple être utilisées par un analyste pour spécifier qu'il est intéressé plus par certaines activités que par d'autres en fonction de leur impact. Par exemple, pour un processus de santé, les activités qui représentent des opérations à risques sont très importantes. Les LPMs extraits représenteraient donc des comportements se déroulant autour de l'exécution de telles activités. Il est à noter que ces activités peuvent avoir un faible nombre d'occurrences, auquel cas elles ne pourraient être contenues dans les LPMs extraits par une extraction "classique" basée sur le support.

Avec les contraintes de profit définies sur le périmètre activité, un analyste peut exiger un certain nombre d'occurrences pour qu'une activité apparaisse dans un LPM. Il peut définir une fonction $f_1^a(activities(\mathcal{L}), activities(\Gamma_{\mathcal{L},LPM}))$ pour indiquer par quelles activités il est intéressé. Il est à noter qu'il est possible d'effectuer un extraction "classique" de LPM basée sur le support [Tax et al., 2016b] en définissant une fonction de profit assignant une importance identique pour chaque activité ; i.e., $f^a(activities(\mathcal{L}), activities(\Gamma_{\mathcal{L},LPM})) = |activities(\Gamma_{\mathcal{L},LPM})|$.

Lors de la phase de construction des LPMs, ajouter une activité dont le profit est nul ne peut jamais améliorer le profit total du LPM. Par conséquent, les fonctions de profit qui assigne un profit nul à certaines activités accélère l'extraction en limitant l'espace de recherche $LPMS(\mathcal{L})$, car les LPMs avec des activités à profit nul ne doivent pas être générés.

Pour l'exemple du processus hospitalier, une contrainte de profit définie sur le périmètre activité peut être utilisée pour représenter la volonté de l'analyste de ne pas se concentrer sur les activités exécutées automatiquement par le système d'information de l'hôpital. Avec $automated\_activities$ l'ensemble des activités exécutées automatiquement par le système d'information, l'analyste peut définir la contrainte suivante : $c^a_{exemple}(activities(\mathcal{L}), activities(\Gamma_{\mathcal{L},LPM})) = \forall act \in activities(\Gamma_{\mathcal{L},LPM}), act \notin automated\_activities$.

6.1.2.4 Contraintes et fonctions de profit définies sur le périmètre modèle

Les contraintes et fonctions de profit définies sur le périmètre modèle peuvent être utilisées lorsque l'analyste a des préférences ou des exigences sur les propriétés structurelles des LPMs. Elles sont de la forme $c^m(LPM)$ et $f^m(LPM)$; i.e., elles portent sur les LPMs seuls et sont indépendantes du log utilisé pour construire les LPMs. Par exemple, si un analyste est intéressé par les comportements menant à l'exécution de l'activité a, il peut définir une contrainte de profit sur le périmètre modèle pour exiger que tous les éléments de $\mathfrak{L}(LPM)$ se terminent par l'activité a.

Généralement, l'analyste sera intéressé par des LPMs qui représentent d'une certaine façon le log. De ce fait, les contraintes et fonctions de profit qui opèrent sur le périmètre modèle sont souvent peu utilisées en tant que telles, mais trouvent leur utilité lorsqu'elles sont combinées avec d'autres contraintes et fonctions définies sur les autres périmètres. $\Gamma_{\mathcal{L},LPM}$ n'a pas besoin d'être calculé pour déterminer si une contrainte sur le périmètre modèle est respectée ou non par un LPM. Par conséquent, ces contraintes peuvent aussi être utilisées pour accélérer l'extraction de LPMs en limitant l'espace de recherche $LPMS(\mathcal{L})$ aux LPMs qui respectent ces contraintes structurelles.

Pour l'exemple du processus hospitalier, l'analyste désire analyser en détail les activités exécutées plusieurs fois par trace. Il peut donc définir une contrainte, notée $c^m_{exemple}$, qui exige que tous les LPMs contiennent une boucle.

6.1.2.5 Fonctions de profit composites

Différentes contraintes et fonctions de profit définies sur plusieurs périmètres peuvent être combinées afin de ne constituer qu'une seule fonction de profit dite *composite*. Par conséquent, il est possible de définir de façon générale le profit d'un LPM par la fonction suivante :

$$u(\mathcal{L}, LPM) = \prod_{i=1}^{n} c_i(\mathcal{L}, LPM) \times \sum_{j=1}^{k} f_j(\mathcal{L}, LPM) \tag{6.1}$$

Le profit total d'un LPM est donc une pondération des différentes fonctions de profit définies sur les différents périmètres. Il est à noter qu'il est possible de donner plus d'importance à certaines fonctions de profit en incluant des poids dans les différentes fonctions f_i.

Dans les sections précédentes, nous avons montré comment nous pouvions modéliser les caractéristiques de l'analyse du processus hospitalier en utilisant les contraintes et fonctions de profit définies sur différents périmètres. Pour résumer, nous avons modélisé :

— une contrainte sur le périmètre trace $c^t_{exemple}$ pour spécifier que les LPMs doivent représenter des comportements exécutés dans une certaine fenêtre de temps

— une fonction sur le périmètre event $f^{e}_{exemple}$ pour assigner un profit à un LPM en fonction du nombre d'heures-homme consommées par les events qu'il peut rejouer

— une contrainte sur le périmètre activité $c^{a}_{exemple}$ pour restreindre l'analyse aux activités non automatisées par le système d'information

— une contrainte sur le périmètre modèle $c^{m}_{exemple}$ pour se concentrer sur les tâches exécutées plusieurs fois par trace

Combinées, ces contraintes et fonctions forment la fonction composite suivante : $u_{exemple}(\mathcal{L}, LPM) = c^{t}_{exemple} \times c^{a}_{exemple} \times c^{m}_{exemple} \times f^{e}_{exemple}$, et modélisent tous les besoins présentés par l'analyste.

Le domaine d'application des contraintes et fonctions de profit est une généralisation de celui présenté dans [Tax et al., 2016b], et les mesures de qualité que les auteurs définissent sont des instanciations des contraintes et fonctions de profit que nous proposons. Par exemple le support est une instanciation de fonction de profit définie sur le périmètre activité, comme nous l'avons présenté dans la sous-section 6.1.2.3. Le déterminisme est un autre exemple de mesure. Inversement proportionnel au nombre moyen de transitions déclenchables pour chaque event dans $\Gamma_{\mathcal{L},LPM}$ rejoué dans le LPM, cette mesure peut être représentée par une fonction de profit composite sur les périmètres trace et modèle.

Il est aussi à noter que les contraintes et fonctions de profit définies sur les périmètres trace et event ne sont pas limitées à l'utilisation d'attributs quantitatifs. Beaucoup de logs contiennent des attributs qualitatifs tel que le *risque* ou *l'impact*, pouvant prendre des valeurs telles que *faible*, *moyen* ou *élevé*. Pour pouvoir utiliser les différents périmètres, il faut au préalable assigner à chaque valeur qualitative possible une valeur quantitative.

Nous présentons dans la Figure 6.3 les différentes étapes de la technique de HU-LPM Discovery.

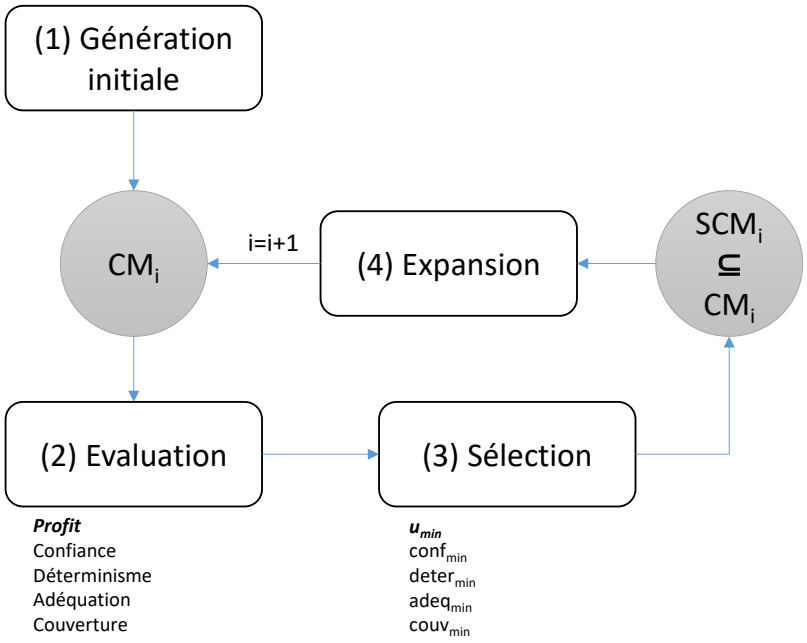


FIGURE 6.3 – Fonctionnement de la technique de HU-LPM Discovery.

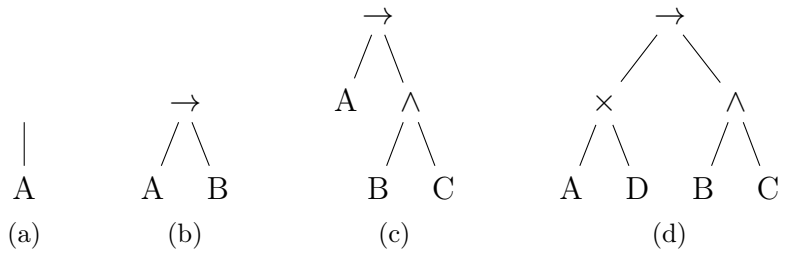


FIGURE 6.4 – *(a)* Un LPM initial LN_1 et *(b)* LN_2, *(c)* LN_3, *(d)* LN_4, trois LPMs générés à partir d'expansions successives.

Définition 33 - Problème d'Extraction de Modèles de Processus Partiels Profitables :
Un LPM est appelé un High-Utility Local Process Models (HU-LPM) dès lors qu'il possède un profit intéressant pour l'analyste qui l'extrait ; i.e., s'il possède un profit supérieur ou égal à un seuil de profit minimum u_{min}, ou s'il fait partie des k LPMs ayant le plus haut profit. Le problème d'extraction de modèles de processus partiels profitables est la tâche qui consiste à extraire l'ensemble des HU-LPMs à partir d'un log $\mathcal{L}$.

6.1.3 Limites de la représentation

Nous avons vu que le problème d'extraction de modèles de processus partiels profitables permettait d'extraire des LPMs intéressants, capables de répondre à des questions métier plus efficacement que ne le fait le problème d'extraction de modèles de processus partiels classique. Cependant, il soulève de nouveaux challenges.

Comme nous l'avons vu, la procédure itérative d'expansion des LPMs est un problème combinatoire qui dépend du nombre d'activités dans le log et du nombre d'activités autorisées dans chaque LPM. Cette caractéristique impacte beaucoup les temps de calculs nécessaires pour extraire un ensemble de LPMs. Pour palier ce problème, la technique de LPM Discovery utilise certaines mesures anti-monotones pour élaguer l'espace de recherche.

Dans nos travaux, nous utilisons la mesure de profit pour évaluer la pertinence d'un LPM. A l'image de la mesure de *coût global* que nous avons définie dans la première partie de ce manuscrit ; et du concept plus général *d'utilité*, la mesure de profit ne possède pas de propriété monotone ou anti-monotone. Définissons $\mathcal{M}$ l'univers des LPMs, i.e., l'ensemble des LPMs qu'il est possible de construire à partir des activités d'un log, et $Exp(LN)$ l'ensemble des LPMs que l'on peut construire à partir d'une expansion de LN. Un LPM $LN \in \mathcal{M}$ peut être étendu en un LPM $LN' \in Exp(LN)$ avec un profit inférieur, égal ou supérieur à celui de LN.

Par exemple, considérons LN_1, LN_2, LN_3 et LN_4 les LPMs présentés respectivement dans les figures 6.4a, 6.4b, 6.4c et 6.4d. Soit le log $\mathcal{L}$ composé d'une seule trace σ présentée dans la Figure 6.5. Le profit d'un LPM est évalué avec une fonction de profit définie sur le périmètre event, $f^e_{exemple}(events(L), events(\Gamma_{L,LPM})) = \sum_{e \in events(\Gamma_{L,LPM})} \pi_{cost}(e)$. Cela donne les résultats suivants : $u(\mathcal{L}, LN_1)$=1350, $u(\mathcal{L}, LN_2)$=2800, $u(\mathcal{L}, LN_3)$=1300 et $u(\mathcal{L}, LN_4)$=1400. De cette façon, il est facile de constater que la mesure de profit n'est pas anti-monotone car $u(\mathcal{L}, LN_3) < u(\mathcal{L}, LN_2)$ et $u(\mathcal{L}, LN_4) > u(\mathcal{L}, LN_3)$.

Élaguer l'espace de recherche ne peut donc pas s'effectuer à l'aide de la mesure de profit. De plus, il n'est pas possible d'utiliser de bornes supérieures telles que celles pré-

event id	activity	time	cost
1	A	26-3-2017 13 :00	100 €
2	D	26-3-2017 13 :27	200 €
3	B	26-3-2017 13 :25	300 €
4	C	26-3-2017 13 :35	100 €
5	D	26-3-2017 13 :42	400 €
6	C	26-3-2017 15 :47	200 €
7	A	26-3-2017 16 :10	100 €
8	C	26-3-2017 16 :34	400 €
9	B	26-3-2017 16 :52	300 €
10	D	26-3-2017 16 :59	200 €
11	A	26-3-2017 17 :13	1000 €
12	B	26-3-2017 17 :15	1000 €
13	A	26-3-2017 17 :16	150 €

FIGURE 6.5 – Exemple de trace.

sentées dans la première partie de ce manuscrit car ces dernières sont uniquement basées sur le périmètre activité du domaine d'application $T.E.A.M$ que nous avons défini. Les contraintes et fonctions de profit du problème d'extraction de modèles de processus partiels profitables peuvent être définies sur une combinaison des 4 périmètres du domaine, le profit d'un event dépend du contexte dans lequel il est rejoué et ne peut se prévoir à l'avance pour définir des bornes supérieures.

Bien que la mesure de profit telle que nous l'avons définie ne possède pas de propriété anti-monotone, nous remarquons des tendances dans l'évolution de cette mesure au fil des expansions des différents LPMs ; i.e., l'évolution n'est pas "anarchique". Une approche d'élagage sans perte pour réduire les temps de calcul n'étant pas possible, nous exploitons dans la section suivante les tendances d'évolution de la mesure de profit au travers d'heuristiques.

6.2 Proposition d'heuristiques pour le problème d'extraction de modèles de processus partiels profitables

Les *heuristiques* sont des méthodes de calcul qui fournissent rapidement une solution réalisable, et sont beaucoup utilisées dans les problèmes d'optimisation difficiles, e.g., dans les problèmes d'optimisation combinatoire, en théorie des graphes ou encore en intelligence artificielle. Le problème d'extraction de LPMs étant un problème combinatoire, les heuristiques sont de bonnes candidates pour réduire les temps de calculs. La contrepartie du gain en temps de calcul est que la solution trouvée n'est pas nécessairement optimale. Pour le problème d'extraction de modèles de processus partiels profitables, cela signifierait que l'ensemble de HU-LPMs extraits ne soit pas exactement l'ensemble optimal ; i.e., il peut exister des LPMs qui ne sont pas dans l'ensemble extrait mais qui ont un profit supérieur à celui de certains LPMs de cet ensemble. Lors de la définition d'une heuristique, l'objectif est d'obtenir le meilleur compromis entre le gain en temps de calcul et la qualité de la solution trouvée.

Dans l'optique de résoudre ce problème, nous proposons la Heuristic HU-LPM Discovery, une extension à la technique de HU-LPM Discovery que nous avons proposée dans la section précédente pour intégrer des approches heuristiques. Ces heuristiques se basent sur l'évolution du profit des LPMs au cours des différentes expansions pour définir le meilleur compromis possible.

Après avoir introduit des concepts de base, nous proposons deux heuristiques *sans mémoire* dans la sous-section 6.2.2 et deux heuristiques *avec mémoire* dans la sous-section 6.2.3.

6.2.1 Concepts

Soit $LN \in \mathcal{M}$ un LPM. $Par(LN)$ représente le parent de LN ; i.e., le LPM à partir duquel LN a été généré, $Par(LN){=}LN' \in \mathcal{M}$ tel que $LN \in Exp(LN')$. Par exemple, pour les process trees présentés dans la Figure 6.4, $Par(LN_3){=}LN_2$. Nous généralisons le concept de parent dans l'Equation 6.2, et définissons $Par^i(LN)$ le i^e parent de LN, avec $Par^0(LN){=}LN$, $Par^1(LN){=}Par(LN)$, $Par^2(LN){=}Par(Par(LN))$ et ainsi de suite. Plus formellement, nous définissons $Par^i(LN)$ comme suit :

$$Par^i(LN) = \begin{cases} Par^{i-1}(Par(LN)) & \text{si } i \geq 1, \\ LN & \text{sinon.} \end{cases} \tag{6.2}$$

Par exemple, pour les process trees présentés dans la Figure 6.4, $Par^3(LN_4){=}LN_1$. Il est à noter que $LN \notin dom(Par)$ pour tout LPM LN qui est un LPM initial, un LPM initial n'étant pas généré à partir d'un autre LPM.

Nous définissons $it\_nb(LN)$ le nombre d'expansions successives nécessaires pour atteindre un LPM LN à partir d'un LPM initial. Formellement, nous définissons $it\_nb(LN)$ comme suit :

$$it\_nb(LN) = \begin{cases} it\_nb(Par(LN)) + 1, & \text{si } LN{\in}dom(Par), \\ 0, & \text{sinon.} \end{cases} \tag{6.3}$$

Pour les process trees présentés dans la Figure 6.4, $it\_nb(LN_3){=}2$.

A partir de Par et it_nb, nous définissons $Anc(LN)$ l'ensemble des *ancêtres* de LN. Nous définissons formellement $Anc(LN)$ comme suit :

$$Anc(LN) = \bigcup_{i=1}^{it\_nb(LN)} Par^i(LN) \tag{6.4}$$

Pour les process trees présentés dans la Figure 6.4, $Anc(LN_4){=}\{LN_1, LN_2, LN_3\}$.

Enfin, nous définissons exp_max le nombre d'expansions successives autorisées à partir d'un LPM initial.

A partir de ces concepts, nous proposons dans les deux sous-sections suivantes quatre heuristiques, deux heuristiques *sans mémoire* et deux heuristiques *avec mémoire* [Dalmas et al., 2017c].

6.2.2 Heuristiques sans mémoire

La première catégorie d'heuristiques que nous proposons sont des heuristiques que nous appelons *"sans mémoire"*. Ces heuristiques se concentrent sur des comparaisons *locales* ; i.e., un LPM est uniquement comparé à son parent direct, les autres expansions étant ignorées. Les heuristiques fonctionnent comme suit : pour un nombre d'expansions successives prédéfini, noté k tel que $0 < k < exp\_max$, un LPM est autorisé à avoir un profit inférieur ou égal à celui de son parent.

Pour chaque heuristique, nous définissons une fonction de *continuité*, $ctn(\mathcal{L}, k, LN)$, dont l'objectif est d'indiquer si un LPM doit être étendu ou non. La fonction $ctn(\mathcal{L}, k, LN)$ vaut 1 si les k plus récentes expansions qui ont mené à la génération du LPM LN à partir du log $\mathcal{L}$ ont satisfait les conditions de l'heuristique. Cela indique que LN peut encore être étendu. Autrement, la fonction $ctn(\mathcal{L}, k, LN)$ vaut 0, indiquant que l'expansion de LN doit être arrêtée. De cette façon, il est possible de réduire l'espace de recherche et donc réduire les temps de calcul nécessaires pour extraire l'ensemble des HU-LPMs. Nous introduisons l'*Heuristique 1* (H_1) comme suit :

Définition 34 - Heuristique 1 (H_1) :
L'expansion d'un LPM $LN \in \mathcal{M}$ est arrêtée si chaque LPM du $k-1^e$ parent de LN jusqu'à LN lui-même a un profit inférieur ou égal à celui de son parent.

$$ctn(\mathcal{L}, k, LN) = \begin{cases} 1 - \prod_{i=0}^{min(it\_nb(LN),k)-1} (u(\mathcal{L}, Par^i(LN)) \leq u(\mathcal{L}, Par^{i+1}(LN))), & \text{si } it\_nb(LN) \geq 1 \\ 1, & \text{sinon.} \end{cases} \quad (6.5)$$

Il est à noter que $u(\mathcal{L}, Par^i(LN)) \leq u(\mathcal{L}, Par^{i+1}(LN)$ est une expression booléenne évaluée à 1 si elle est vrai et 0 sinon. Comme nous voulons que les LPMs initiaux soient toujours étendus indépendamment de l'heuristique utilisée, $ctn(\mathcal{L}, k, LN)$=1 lorsque $it\_nb(LN)$=0.

Nous proposons aussi l'*Heuristique 2* (H_2), une version assouplie de l'Heuristique 1. Dans cette version, un LPM est autorisé à avoir un profit identique à celui de son parent.

Définition 35 - Heuristique 2 (H_2) :
L'expansion d'un LPM $LN \in \mathcal{M}$ est arrêtée si chaque LPM du $k-1^e$ parent de LN jusqu'à LN lui-même a un profit strictement inférieur à celui de son parent.

$$ctn(\mathcal{L}, k, LN) = \begin{cases} 1 - \prod_{i=0}^{min(it\_nb(LN),k)-1} (u(\mathcal{L}, Par^i(LN)) < u(\mathcal{L}, Par^{i+1}(LN))), & \text{si } it\_nb(LN) \geq 1 \\ 1, & \text{sinon.} \end{cases} \quad (6.6)$$

6.2.3 Heuristiques avec mémoire

Le second type d'heuristiques que nous proposons sont des heuristiques que nous appelons *"avec mémoire"*. Ces heuristiques "retiennent" les différents LPMs qui ont été générés. Au lieu de comparer deux expansions successives, ces heuristiques comparent un LPM avec son meilleur ancêtre. Pour un LPM LN, nous définissons

$B(\mathcal{L}, LN)$ profit du meilleur ancêtre de LN ; i.e., le profit le plus élevé parmi les ancêtres de LN dans le log $\mathcal{L}$; $B(\mathcal{L}, LN){=}u(\mathcal{L}, LN')$ avec $LN'{\in}Anc(LN)$ tel que $\nexists LN''{\in}Anc(LN)|u(\mathcal{L}, LN''){>}u(\mathcal{L}, LN')$. Les heuristiques fonctionnent comme suit : pour un nombre d'expansions successives prédéfini, noté k tel que $0 < k < exp\_max$, un LPM est autorisé à avoir un profit inférieur ou égal à celui de son meilleur ancêtre. Nous introduisons l'*Heuristique 3* (H_3) comme suit :

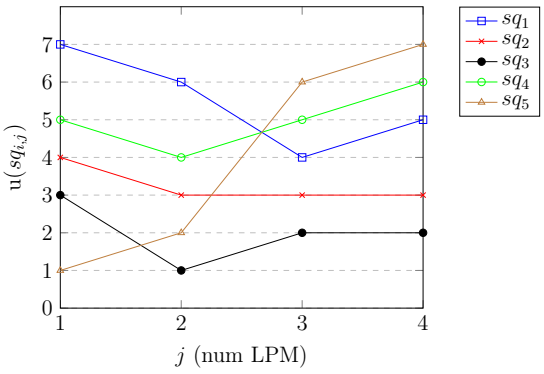
Définition 36 - Heuristique 3 (H_3) :
L'expansion d'un LPM $LN \in \mathcal{M}$ est arrêtée si chaque LPM du $k{-}1^e$ parent de LN jusqu'à LN lui-même a un profit inférieur ou égal à celui de son meilleur ancêtre.

$$ctn(\mathcal{L}, k, LN){=}\begin{cases} 1{-}\prod_{i=0}^{min(it\_nb(LN),k){-}1} (u(\mathcal{L}, Par^i(LN)){\leq}B(\mathcal{L}, Par^i(LN))), & \text{si } it\_nb(LN){\geq}1 \\ 1, & \text{sinon.} \end{cases} \quad (6.7)$$

Nous proposons aussi l'*Heuristique 4* (H_4), une version assouplie de l'Heuristique 3. Dans cette version, un LPM est autorisé à avoir un profit identique à celui de son meilleur ancêtre.

Définition 37 - Heuristique 4 (H_4) :
L'expansion d'un LPM $LN \in \mathcal{M}$ est arrêtée si chaque LPM du $k{-}1^e$ parent de LN jusqu'à LN lui-même a un profit strictement inférieur à celui de son meilleur ancêtre.

$$ctn(\mathcal{L}, k, LN){=}\begin{cases} 1{-}\prod_{i=0}^{min(it\_nb(LN),k){-}1} (u(\mathcal{L}, Par^i(LN)){<}B(\mathcal{L}, Par^i(LN))), & \text{si } it\_nb(LN){\geq}1 \\ 1, & \text{sinon.} \end{cases} \quad (6.8)$$



(a)

exp. seq. \ heuristic	H_1	H_2	H_3	H_4
sq_1 :				
sq_2 :		✓		
sq_3 :	✓	✓		
sq_4 :	✓	✓		✓
sq_5 :	✓	✓	✓	✓

(b)

FIGURE 6.6 – *(a)* Quatre expansions successives à partir de cinq LPMs initiaux et *(b)* le périmètre d'élagage des différentes heuristiques.

Pour illustrer le fonctionnement des différentes heuristiques proposées, nous présentons un exemple dans la Figure 6.6a. Elle représente l'évolution du profit pour différents LPMs générés à partir d'expansions successives, à partir de LPMs initiaux.

Soit sq_i une séquence de 3 expansions depuis un LPM initial, et $sq_{i,j}$ le j^e LPM de la séquence sq_i. Pour chacune des 4 heuristiques et k=2, la Figure 6.6b indique les séquences qui seraient autorisées à aller jusqu'à la 3^e expansion (annotées de ✓), et celles qui auraient été arrêtées avant.

Dans cet exemple, sq_1 et sq_5 sont deux extrêmes. Alors que sq_1 contient deux diminutions successives du profit ($sq_{1,2}$ et $sq_{1,3}$), sq_5 voit le profit augmenter à chaque expansion. Par conséquent, toutes les heuristiques auraient arrêté sq_1 après $sq_{1,3}$; supprimant les expansions suivantes de l'espace de recherche, et auraient laissé sq_5 s'étendre jusqu'à $sq_{5,4}$.

sq_2 est seulement étendue jusqu'à la fin par H_2 car cette version assouplie permet que le profit des LPMs générés stagne sur deux expansions successives ($sq_{2,3}$ et $sq_{2,4}$).

A l'inverse, l'expansion de sq_4 est seulement arrêtée par H_3 car le profit des deux expansions successives $sq_{4,2}$ et $sq_{4,3}$ est au mieux identique à celui de $sq_{3,1}$.

Enfin, sq_3 est étendue jusqu'à la fin par les heuristiques sans mémoire, mais arrêtée avant par les heuristiques avec mémoire car l'augmentation visible à $sq_{3,3}$ est suffisante pour que le profit de l'expansion soit supérieur à celui de son parent, mais pas assez pour qu'il soit au moins égal à celui de son meilleur ancêtre.

L'heuristique H_2 est la plus permissive car un LPM est seulement comparé à son parent et est autorisé à avoir un profit identique. A l'inverse, l'heuristique H_3 est la plus restrictive car elle exige qu'un LPM ait un profit supérieur à celui de son meilleur ancêtre. Comme nous l'avons dit, les heuristiques ne donnent pas nécessairement de solutions optimales ; i.e., il se peut que certains bons LPMs ne soient pas extraits. C'est le cas dans l'exemple de la séquence sq_4 qui auraient été arrêtée par H_3 après $sq_{4,3}$. Nous remarquons que le LPM généré par l'expansion suivante a un profit supérieur à celui de son meilleur ancêtre. Ceci est un exemple d'expansion qui n'aurait pas dû être arrêtée.

Les différentes étapes de la technique de Heuristic HU-LPM Discovery sont résumées dans la Figure 6.7.

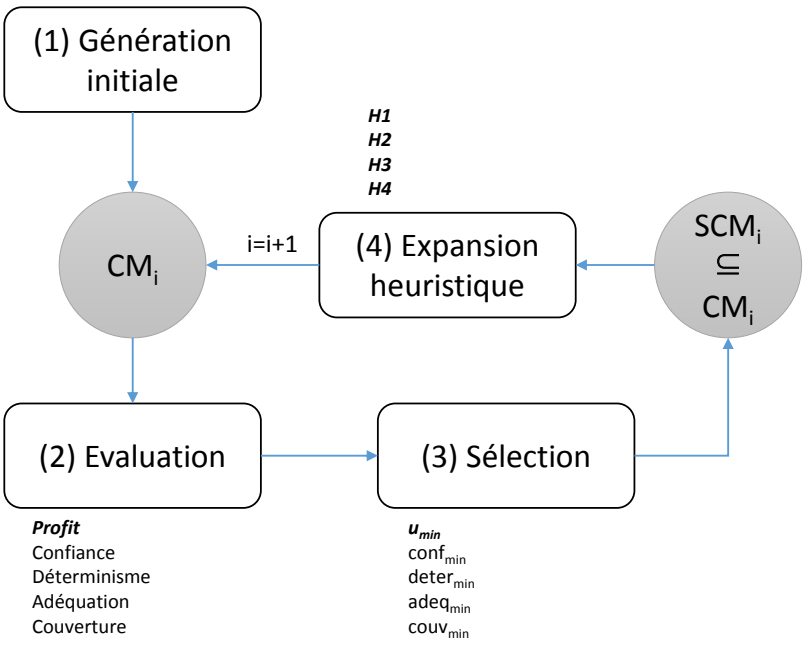


FIGURE 6.7 – Fonctionnement de la technique de Heuristic HU-LPM Discovery.

Conclusion

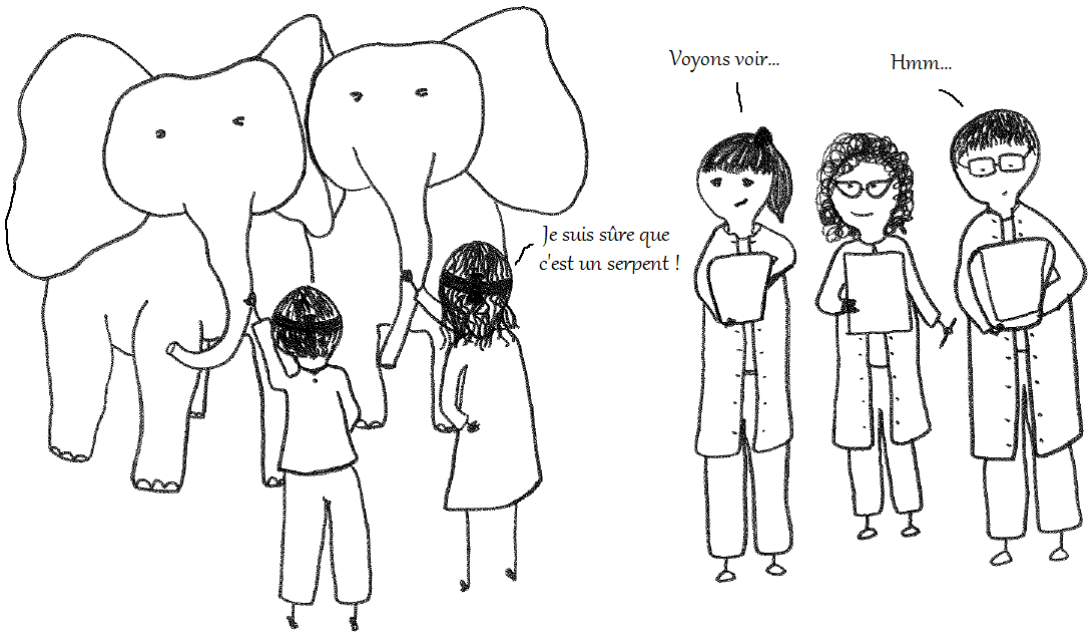
L'objectif de ce chapitre était de proposer une nouvelle approche pour modéliser les comportements intéressants dans des processus peu structurés. Pour cela, nous avons formalisé le problème d'extraction de modèles de processus partiels profitables. L'objectif de ce problème est d'extraire, à partir d'un event log, l'ensemble des HU-LPMs ; i.e., l'ensemble des fragments de comportements jugés intéressants par l'analyste qui les génère. Afin de permettre ce type d'extraction, nous avons proposé une nouvelle technique, la HU-LPM Discovery, ainsi qu'un ensemble de contraintes et fonctions de profit afin d'utiliser la richesse des event log pour définir l'intérêt des différents LPMs générés.

Nous avons vu que l'extraction de HU-LPMs est un problème combinatoire, de plus la mesure de profit utilisée ne permet pas de réduire l'espace de recherche tout en garantissant une extraction optimale. Par conséquent, nous avons exploré la piste des méthodes approchées en proposant la technique de Heuristic HU-LPM Discovery, dont l'objectif est de réduire les temps de calcul au détriment de la garantie d'extraire tous les HU-LPMs.

Chapitre 7

Mise en œuvre des propositions

Sommaire



Introduction

Dans le chapitre précédent, nous avons formalisé le problème d'extraction de modèles de processus partiels profitables. Pour résoudre ce problème, nous avons proposé un ensemble de contraintes et fonctions de profit pour mesurer l'intérêt des différents modèles générés. Ensuite, nous avons proposé plusieurs heuristiques, des méthodes approchées pour réduire les temps de calcul nécessaires mais qui ne garantissent pas l'extraction de tous les HU-LPMs. L'objectif de ce chapitre est d'évaluer les performances des approches proposées. Toutes les approches développées dans la seconde partie de ce manuscrit ont été implémentées dans l'outil ProM [1], et font partie du package *LocalProcessModelDiscovery* [2].

Ce chapitre est organisé comme suit. La section 7.1 est dédiée à l'application du problème d'extraction de modèles de processus partiels profitables. Pour cela, nous appliquons les approches développées sur quatre event logs.

Dans la section 7.2, nous évaluons la performance des différentes heuristiques proposées sur des event logs issus de la littérature et sur un ensemble d'event logs générés artificiellement. Pour tenter de détecter des relations entre la performance des heuristiques et les caractéristiques des events logs, nous explorons certaines techniques d'apprentissage supervisé.

7.1 Applications du problème d'extraction de modèles de processus partiels profitables

Dans cette section, nous montrons l'intérêt du problème d'extraction de modèles de processus partiels profitables en l'appliquant sur quatre event logs. Les trois premiers utilisent des event logs issus de la littérature. L'event log utilisé dans la première étude de cas ayant aussi été utilisé par d'autres techniques similaires à la notre, nous présentons une comparaisons des informations extraites. Le dernier event log utilisé est celui construit à partir des données du Centre Hospitalier de Craponne-sur-Arzon (CHPCA). Ces études de cas représentent quatre domaines différents, l'objectif étant de montrer que même si nos travaux sont motivés par les processus de santé, les approches que nous proposons sont génériques et applicables à différents domaines.

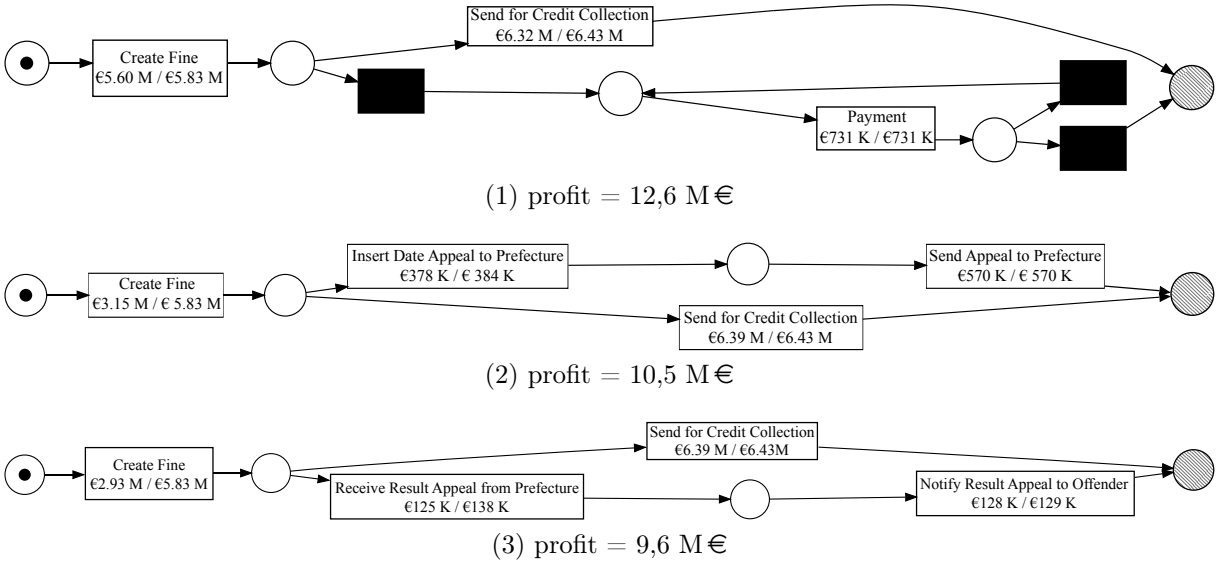
7.1.1 Gestion d'amendes pour infractions routières

L'event log de gestion d'amendes pour infractions routières est un log dans lequel chaque trace représente la gestion d'une amende [3]. Chaque trace commence avec l'émission de l'amende (*create fine*), qui a un attribut *amount* représentant le montant de l'amende. Cet event a aussi un attribut *dismissal* représentant une ou plusieurs raisons de rejet de l'amende. Cet attribut prend la valeur *"NIL"* si l'amende doit absolument être payée. Un paiement (*payment*) a un attribut *paymentAmount* indiquant le montant de l'amende payé. Le montant total de l'amende peut être payé en plusieurs fois. Certaines amendes sont payées directement à l'officier de police lorsque l'amende est donnée, d'autres sont envoyées par courrier (*send fine*). Un envoi postal a un attribut (*expense*) représentant le

1. http ://www.promtools.org/doku.php
2. https ://svn.win.tue.nl/trac/prom/browser/Packages/LocalProcessModelDiscovery
3. https ://doi.org/10.4121/uuid :270fd440-1057-4fb9-89a9-b699b47990f5

coût de gestion du dossier, s'ajoutant au montant total de l'amende à payer. Lorsqu'une amende n'est pas payée à temps, une pénalité de retard (*add penalty*) est appliquée, avec un attribut (*amount*) qui indique le montant de la pénalité à ajouter au montant de l'amende. Si l'amende n'est toujours pas payée après l'application de la pénalité de retard, elle est envoyée en recouvrement (*send for credit collection*). De plus, une amende peut être contestée à la préfecture (*send appeal to prefecture*), voir même devant les tribunaux (*appeal to court*). Si la contestation est concluante, l'attribut *dismissal* de l'event *create fine* prend la valeur *"G"* ou *"#"*.

Au total, cet event log contient 11 activités distinctes et 150 370 traces d'une longueur moyenne de 4 events ; la trace la plus courte et la trace la plus longue comportant respectivement 2 et 20 events.



(1) profit = 12,6 M€

(2) profit = 10,5 M€

(3) profit = 9,6 M€

FIGURE 7.1 – Les trois HU-LPMs avec le plus haut profit extraits à partir du log de gestion des amendes pour infractions routières, en utilisant *le montant restant à payer* comme profit.

Supposons un analyste avec le besoin d'analyser les parties du processus de gestion des amendes où le montant restant à payer est élevé.

Pour atteindre cet objectif, nous utilisons une fonction de profit qui définit le profit comme le montant de l'amende restant à payer au moment où l'event est rejoué. Cette fonction est définie sur le périmètre trace car elle dépend, pour un γ-segment rejoué dans le LPM, du dernier attribut "*amount*" rencontré (soit dans un event "*create fine*" soit dans un event "*add penalty*"), plus le potentiel surplus dû à un envoi postal, moins le montant des paiements déjà effectués.

L'extraction des HU-LPMs sur cet event log prend 39 minutes, sur une machine équipée d'un processeur Intel i7 2,4GHz et 16Go de RAM.

Nous présentons dans la Figure 7.1 les 3 meilleurs HU-LPMs ; i.e., les 3 LPMs avec le profit le plus élevé. De façon presque identique à un LPM, un HU-LPM est modélisé par un réseau de Petri dont les transactions sont sous la forme ($actX/Y$), avec *act* le label de l'activité modélisée, mais avec Y le profit total des events dans le log ayant exécuté *act*, et X le profit total des events de Y rejouables dans le LPM.

Le premier HU-LPM montre qu'au total, 5,83 M€ d'amendes ont été émises. Parmi ces amendes, certaines ont été envoyées en recouvrement, d'autres ont reçu des paiements,

le tout pour un montant de 5,60 M€. Les 230 K€ restant correspondent aux amendes qui n'ont pas encore été payées mais qui sont assez récentes pour ne pas avoir été encore envoyées en recouvrement. Nous remarquons que le montant total des paiements reçus après un event *create fine* est de 731 K€, ce qui est plutôt faible comparé aux 5,83 M€ qui sont dus. Il est à noter que les amendes envoyées en recouvrement représentent un montant total de 6,43 M€, ce qui est supérieur au montant total initial de 5,83 M€. Cet écart s'explique par l'ajout de pénalités de retard, et dans une moindre mesure, des frais additionnels pour envoi postal. Enfin, des 6,43 M€ d'amendes envoyées en recouvrement, la plupart d'entre elles, d'un montant de 6,32 M€, représentent des events rejouables dans le premier HU-LPM ; i.e., l'envoi en recouvrement apparaît après la création de l'amende et est dans une situation de choix exclusif avec le paiement de l'amende. Puisque nous savons que chaque trace commence avec la création de l'amende, la seule explication possible est que le montant de 6,32 M€ d'amendes envoyées en recouvrement n'a pas reçu un seul paiement avant l'envoi, alors qu'une partie des amendes restantes aussi envoyées en recouvrement, d'un montant de 110 K€, ont reçu au moins un premier paiement partiel.

Le second HU-LPM présenté dans la Figure 7.1 montre que sur un total de 5,83 M€, 3,15 M€ a soit été envoyé en recouvrement soit contesté en préfecture. Une contestation est d'abord enregistrée (*insert date appeal to prefecture*) avant d'être envoyée en préfecture (*send appeal to prefecture*). Le HU-LPM montre que, pour les amendes contestées, le montant total est de 378 K€ lors de l'enregistrement, mais les pénalités pour retard de paiement fait passer ce montant à 570 K€ lors de l'envoi en préfecture. Cela signifie qu'en moyenne, le montant des amendes sont multipliées par un facteur 1,5 lors de la procédure de contestation en préfecture à cause des retards de paiement générés. Enfin, le montant des amendes envoyées en recouvrement montre que seulement une petite partie des amendes créées ont été suivies par une contestation en préfecture.

Le troisième HU-LPM présenté dans la Figure 7.1 modélise un fragment similaire au deuxième HU-LPM, avec la différence qu'il décrit maintenant les étapes suivantes de la contestation en préfecture. Lorsque le verdict de la contestation est reçu, le montant total encore dû est de 138 K€. Ce montant est bien inférieur aux montants respectifs de 378 K€ et 570 K€ des étapes précédentes de la procédure de contestation en préfecture. Cela signifie que sur 570 K€ d'amendes contestées en préfecture, 432 K€ d'amendes n'ont pas eu de verdict suite à la contestation. Les contestations ont donc été soit retirées avant d'être étudiées ou sont encore en attente de verdict. De plus, sur 138 K€ d'amendes contestées en préfecture et dont le verdict a été reçu, 125 K€ représentent des events rejouables dans le troisième HU-LPM, signifiant que les 13 K€ restants représentent des contestations qui n'ont pas encore été notifiées, ou dont les amendes concernées ont été envoyées en recouvrement avant d'être contestées.

Cet event log a déjà été utilisé pour l'application de trois autres techniques existantes, et qui utilisent les attributs disponibles dans l'event log pour en extraire des connaissances : le *MP-Declare Miner* [Schönig et al., 2016b], une technique qui se rapproche de la nôtre et que nous avons analysée dans le chapitre 5 ; et les *Multi-Perspective Process Explorer* [Mannhardt et al., 2015] et *Data-Aware Heuristic Miner* [Mannhardt et al., 2017] qui s'éloignent de notre problématique dans la mesure où elles analysent un modèle sur un périmètre global plutôt que local.

Avec le MP-Declare Miner, les informations suivantes ont pu être extraites : *"(1) Dans 74 % des cas où une pénalité pour retard est donnée, l'amende a été envoyée en recouvrement" ; "(2) Lorsqu'une pénalité est donnée, le montant de cette pénalité est compris*

entre 470€ et 795€"; et "(3) Plus la pénalité est élevée, plus la probabilité que l'amende ne soit pas payée est basse". Il est à noter que ces informations sont différentes de celles que l'on peut obtenir avec des HU-LPMs : alors que les informations extraites par le MP-Declare Miner sont des corrélations entre le control-flow et les attributs ; i.e., entre l'exécution d'une activité et la valeur des attributs au moment de l'exécution ; les HU-LPMs montrent aussi l'évolution de la valeur des différents attributs entre les différentes étapes du control-flow ; e.g. l'évolution des montants restant à payer pour les amendes. Les corrélations entre les attributs et les étapes du control-flow pourraient être facilement extraites à partir des HU-LPM en utilisant des techniques classiques de *decision mining* [Mannhardt et al., 2016].

Avec le Data-Aware Heuristic Miner, les informations suivantes ont pu être extraites : *"(1) L'ajout de pénalités pour retard de paiement dépend de la valeur de l'attribut "dismissal" de l'amende concernée. S'il vaut "G", alors aucune pénalité n'est ajoutée, alors que s'il vaut "NIL" une pénalité est ajoutée et dont la valeur dépend du montant de l'amende"; "(2) Les amendes non payées dont le montant est inférieur à 35€ reçoivent une pénalité"; "(3) Le processus se termine par la notification à l'officier de police du verdict de la contestation de l'amende en préfecture pour les amendes dont l'attribut "dismissal" a la valeur "#" ou "G"."*. La seule information extraite depuis le Multi-Perspective Process Explorer est : *"L'envoi en recouvrement survient pour les amendes dont le montant est supérieur à 71€"*. A l'image des informations extraites du ML-Declare Miner, celles extraites par le Data-Aware Heuristic Miner et le Multi-Perspective Process Explorer concernent toutes des relations entre les attributs et le control-flow. Encore une fois, la différence majeure qu'il existe avec l'extraction de HU-LPM est que cette dernière permet d'extraire des informations sur l'évolution des attributs dans les différentes étapes du control-flow modélisé.

7.1.2 Gestion de tickets informatiques

L'event log de gestion de tickets informatiques est un log mis à disposition lors du *Business Process Intelligence Challenge 2014* (BPI'14)[4]. Il est composé d'incidents liés à des perturbations du service IT d'une institution financière. Chaque *incident* est associé à une ou plusieurs *interactions* représentant les appels et e-mails reçus par les agents du service IT par rapport à l'incident. Lorsqu'un incident est notifié, il est assigné à un opérateur, qui soit le prend en charge et le résout, soit le réassigne à un autre opérateur plus qualifié. Pour chaque incident, plusieurs attributs sont enregistrés, tels que :

— **Service Component WBS** : le numéro d'identification du composant concerné par l'incident.

— **Configuration Item** : le type du composant concerné par l'incident ; e.g., ordinateur, serveur, application. Chaque *service component WBS* appartient à un seul *configuration item*.

— **Impact** : l'impact de la perturbation sur l'activité du client ayant rapporté l'incident. La valeur de l'impact va de 1 à 5.

— **Closure code** : un code qui classe la cause de la perturbation ; e.g., erreur utilisateur (*user error*), problème logiciel (*software error*) ou problème matériel (*hardware error*).

4. https ://doi.org/10.4121/uuid :c3e5d162-0cfd-4bb0-bd82-af5268819c35

— **Causedby** : représente le composant à l'origine de la perturbation du composant concerné par l'incident.
— **Number of interactions** : le nombre d'appels et e-mails reçus par le service IT relatifs à l'incident.
— **Number of reassignements** : le nombre de fois où l'incident a été réassigné d'un opérateur à un autre.

Nous avons transformé le log pour grouper les events en fonction de leur numéro de *service component WBS*. Une trace est donc composée des incidents dans lesquels un composant est impliqué. Nous définissons le label des incidents par la concaténation des attributs *closure code* et *causedBy*, séparé par le caractère '|'. Le log généré contient 944 activités distinctes et 313 traces d'une longueur moyenne de 93 events, la trace la plus petite et la trace la plus longue ayant respectivement 2 et 3595 events.

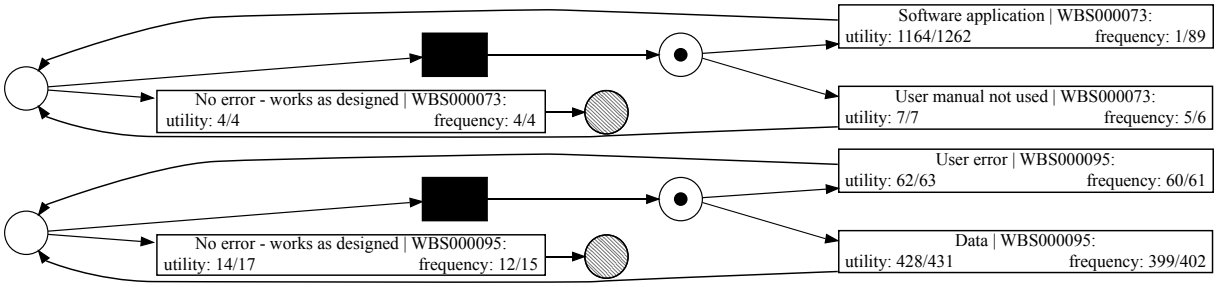


FIGURE 7.2 – Les deux LPMs avec le plus haut profit extraits depuis le log BPI'14 en utilisant l'attribut *number of interactions* dans la fonction de profit.

Supposons un analyste avec le besoin d'analyser les fragments de comportements responsables d'un grand nombre d'appels et d'emails reçus par le service IT. Pour répondre à cette question, nous formulons la fonction de profit suivante : $f^e(events(\mathcal{L}), events(\Gamma_{\mathcal{L},LPM})) = \sum_{g \in events(\Gamma_{\mathcal{L},LPM})} \pi_{number\_of\_interactions}(g)$. L'extraction des HU-LPMs sur cet event log prend 3 minutes, sur une machine équipée d'un processeur Intel i7 2,4GHz et 16Go de RAM.

Nous présentons dans la Figure 7.2 les deux HU-LPMs avec les plus hauts profits extraits à partir de ce log. Le premier HU-LPM a un profit de 1175, indiquant que les fragments de comportement représentés par ce HU-LPM expliquent 1175 des appels et e mails reçus par le service IT. Au total, 1262 appels et e-mails reçus sont associés à des incidents avec un *closure code* lié à une application logicielle (*software application*) et causés par le composant $WBS000073$, parmi lesquels 1164 représententent des events rejouables dans le LPM. Dans le log, il y a 89 incidents avec ce *closure code* et causé par le même composant, et un seul d'entre eux est rejouable dans ce HU-LPM. Cependant, il est à lui seul responsable des 1164 interactions sur les 1262 observées. Enfin, le HU-LPM se termine avec un incident ayant un *closure code* indiquant que le composant n'a pas d'erreur et fonctionne correctement (*No error - works as designed*). Cela signifie que l'application possède une fonctionnalité qui marche normalement d'après le service IT mais perçue comme défaillante par les utilisateurs de l'institution, causant beaucoup de trafic dans le système de ticketing du service IT. Parce qu'un seul incident *software application | WBS000073* est rejouable dans le HU-LPM, nous pourrions dire que le HU-LPM décrit une anomalie plus qu'un comportement. Il est à noter que l'extraction de ce type d'anomalies est le résultat de la fonction de profit que nous avons définie, qui permet l'analyse de profits inégalement répartis entre les différents events.

Le second HU-LPM montre un fragment de comportement similaire , mais concerne les incidents causés par l'application logicielle d'un serveur $WBS000095$. Ce HU-LPM a un profit de 504, ce qui signifie qu'il décrit au total 504 appels et e-mails reçus par le service IT. Le HU-LPM montre que sur 61 erreurs utilisateur (*user errors*), 60 sont éventuellement suivies d'un incident avec un *closure code* indiquant que le composant n'a pas d'erreur et fonctionne correctement (*No error - works as designed*). Ces 60 incidents ensemble génèrent 62 interactions avec le service IT. Il est à noter que 399 des incidents liés à cette application logicielle, responsable de 428 interactions avec le service IT, se terminent aussi avec un incident avec le même *closure code*.

Dans les deux HU-LPMs présentés, nous constatons que deux composants, le $WBS000073$ et le $WBS000095$, sont responsable de la majorité des appels et e-mails reçus par le service IT. De plus, pour chacun d'entre eux, il se trouve que la perturbation rapportée n'en est pas une, ce qui indique que de tels incidents pourraient être empêchés.

7.1.3 Étude comportementale à partir de capteurs

Le process mining a récemment étendu ses applications, initialement tournées vers l'analyse de processus métier, à d'autres domaines allant de l'analyse logicielle [Leemans and van der Aalst, 2015], à l'analyse de comportements humains [Leotta et al., 2015,Sztyler et al., 2015, Tax et al., 2016a, Tax et al., 2016c]. Les applications de techniques de process mining pour l'analyse de comportements humains sont majoritairement axées autour de la génération de modèle sans sémantique formelle [Leotta et al., 2015,Sztyler et al., 2015], les techniques utilisant des notations avec une sémantique formelle nécessitent une phase de pré-processing du log importante pour permettre l'extraction de modèles représentatifs [Tax et al., 2016a, Tax et al., 2016c]. Dans cette étude de cas, nous nous concentrons sur l'analyse de fragments de comportements humains en utilisant des contraintes permettant d'extraire des modèles avec une sémantique formelle sans avoir recours à une phase de pre-processing importante.

[Tapia et al., 2004] ont collecté des données issues de capteurs domotiques installés dans deux maisons, dans l'optique d'analyser les activités de la vie quotidienne. (ADL [Katz, 1983]). Chaque trace contient une journée de données collectées, dans lesquels les events représentent le déclenchement d'un capteur, l'activité exécutée ayant le label du capteur déclenché. Sur les deux event logs collectés, $MITA$ et $MITB$, notre étude porte sur $MITA$. Ce log est composé de 72 activités distinctes et 16 traces d'une longueur moyenne de 173 events, la trace la plus petite et la trace la plus longue ayant respectivement 71 et 482 events.

activity	time
Exhaust fan	04-04-2003 07 :23 :14
Kitchen drawer 3	04-04-2003 11 :07 :18
Kitchen drawer 3	04-04-2003 13 :48 :39
Kitchen drawer 3	04-04-2003 14 :52 :46
Kitchen drawer 3	04-04-2003 17 :49 :08
Kitchen drawer 3	04-04-2003 17 :49 :15
Kitchen drawer 3	04-04-2003 17 :52 :13

TABLEAU 7.1 – Les events *exhaust fan* et *kitchen drawer 3* dans la trace du 4 avril 2003 extrait du log $MITA$.

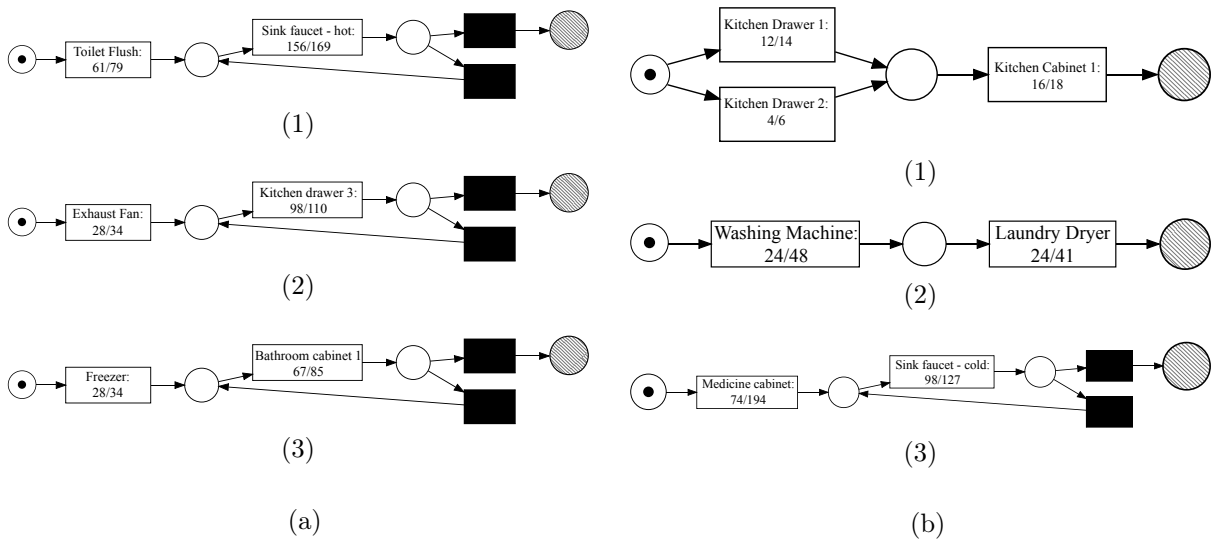


FIGURE 7.3 – *(a)* Les trois LPMs les plus fréquents, et *(b)* les trois HU-LPMs avec le plus haut profit extraits en définissant des contraintes de temps.

Nous présentons dans la Figure 7.3a les 3 meilleurs LPMs extraits par la technique classique de LPM Discovery. Le premier LPM montre que sur 79 fois où la chasse d'eau a été tirée (*toilet flushes*), 61 sont suivies le même jour par de l'eau chaude s'écoulant du robinet de l'évier (*sink faucet - hot*). Il montre aussi que sur les 169 fois où quelqu'un s'est servi de l'eau chaude depuis le robinet de l'évier, 156 étaient à la suite d'un tirage de chasse d'eau.

Le second LPM montre que sur les 34 déclenchements du ventilateur d'extraction (*exhaust fan*), 28 sont suivis d'un comportement répétitif lié à d'ouverture du 3^{e} tiroir de la cuisine (*kitchen drawer 3*). En moyenne, chaque déclenchement du ventilateur est suivi de 3,5 (98/28) ouvertures de tiroir le même jour.

Le troisième LPM montre que sur 34 ouvertures du congélateur (*freezer*), 28 sont suivies de l'ouverture du premier meuble de la salle de bain (*Bathroom cabinet 1*).

Alors que le premier LPM semble avoir du sens, étant donné que les personnes se lavent habituellement les mains après être allées aux toilettes, les comportements décrits par les deux autres LPMs ne semblent pas avoir de corrélations logiques. Nous présentons dans le Tableau 7.1 les events *exhaust fan* et *kitchen drawer 3* sur une même journée du log $MITA$. Pris ensemble, ces events constituent un γ-segment rejouable dans le deuxième LPM. Cependant, l'intervalle de temps entre les events semble être assez important pour qu'il n'y ait aucune relation entre l'event *exhaust fan* à 07 :23 et l'event *kitchen drawer 3* à 11 :07.

Nous avons soulevé dans la première partie de ce manuscrit le problème relatif aux events corrélés mais indépendants car trop espacés dans le temps. Nous pouvons pallier ce problème en définissant un intervalle de temps que deux events consécutifs doivent respecter ; et une fenêtre de temps maximum dans laquelle un γ-fragment doit apparaître. Nous présentons dans la Figure 7.3b les trois meilleurs HU-LPMs extraits en définissant une contrainte de profit exigeant un temps de 2 minutes entre l'exécution de deux activités, et une durée maximum d'exécution pour un γ-fragment de 5 minutes.

Le premier HU-LPMs montre que les events *kitchen drawer 1* et *kitchen drawer 2* sont presque toujours suivis de l'event *kitchen cabinet 1* dans les 2 minutes. Lorsque nous

regardons de plus près les γ-segments rejouables dans ce HU-LPM, nous constatons que ces events se produisent entre 17 :00 et 19 :00. Le HU-LPM semble décrire le comportement de quelqu'un faisant la cuisine, étant donné qu'il y a 16 γ-segments pour 16 traces (et donc 16 jours). Par conséquent, ce HU-LPM peut être utilisé comme un "détecteur" du comportement *"faire la cuisine"*.

Le deuxième HU-LPM montre que la moitié des events relatifs à la machine à laver (*washing machine*) sont suivis dans les deux minutes d'events liés au sèche-linge (*laundry dryer*). Cette moitié représente les fois où la machine à laver a été ouverte dans l'optique de mettre le linge dans le sèche-linge ; l'autre moitié pouvant être les fois où la machine a été ouverte dans le but d'y mettre du linge à laver.

Le troisième HU-LPM montre que l'ouverture de l'armoire à pharmacie (*medecine cabinet*) est suivie dans presque la moitié des cas par quelqu'un se servant de l'eau froide au robinet de l'évier (*sink faucet - cold*), certains events *medecine cabinet* étant suivis d'une séquence d'events *sing faucet - cold*. Bien que les deuxième et troisième HU-LPMs décrivent des comportements routiniers, les trois HU-LPMs extraits montrent que la définition de contraintes de temps permet d'extraire des fragments de comportements dont les events sont corrélés les uns aux autres.

L'extraction de LPM est un problème combinatoire qui dépend en partie du nombre d'activités dans le log. Par conséquent, procéder à l'extraction de LPMs à partir du log $MITA$ avec l'ensemble des 72 activités n'est pas possible. Dans [Tax et al., 2016d], une approche est proposée pour permettre l'extraction de LPMs à partir de ce type d'event log. Cette approche choisit de manière heuristique d'utiliser des sous-ensembles d'activités pour combiner les résultats a posteriori. En utilisant cette méthode, extraire l'ensemble des LPMs sans contrainte de temps prend 5 minutes sur une machine équipée d'un processeur Intel i7 2,4GHz et 16Go de RAM L'extraction des HU-LPMs sur cet event log prend moins d'une minute, sur une même machine ; montrant qu'en plus d'extraire des LPMs plus intéressants, la définition de contraintes et fonctions de profit permet aussi dans certains cas de réduire les temps d'exécution.

7.1.4 Le Centre Hospitalier du Pays de Craponne-sur-Arzon

Dans cette étude de cas, nous reprenons l'event log généré à partir de données extraites du système d'information du Centre Hospitalier du Pays de Craponne-sur-Arzon (CHPCA). Nous avons conclu l'analyse de ce log dans la première partie du manuscrit en évoquant la limite de l'instanciation PRISM à gérer l'aspect "dynamique" du profit ; i.e., le fait que le profit d'un event ne soit pas fixe mais dépende du motif dans lequel il est rejouable. Nous avons poursuivi en déclarant qu'une analyse telle que celle qui consiste à découvrir les motifs représentant une réduction de l'autonomie des patients au cours de leurs différents séjours n'est pas possible. Nous prouvons dans la suite de ce cas d'étude qu'à l'aide de contraintes et fonctions de profit, cette analyse est dorénavant possible.

Pour effectuer notre analyse, nous reprenons le log tel que présenté dans la Partie I. En plus des attributs déjà disponibles, nous avons pour chaque séjour accès aux attributs *gir_in* et *gir_out*, représentant respectivement le niveau de dépendance GIR à l'entrée du CHPCA et le niveau de dépendance GIR à la sortie du CHPCA. L'analyse que nous effectuons vise à extraire les fragments de comportements dans lesquels la dégradation du niveau de GIR est la plus forte. Le niveau GIR est compris sur l'intervalle [1 ;6], avec 1 le niveau de dépendance le plus fort ; i.e., le patient a perdu toute autonomie motrice

et mentale, et 6 le niveau de dépendance le plus faible ; i.e., le patient est entièrement autonome dans les tâches de la vie courante.

Pour cela, nous définissons la fonction de profit $u_{chpca}(\mathcal{L}, LPM) = c_1^t(\mathcal{L}, LPM) \times f_1^e(events(\mathcal{L}), events(\Gamma_{\mathcal{L},LPM})$, définies par les Équations suivantes :

$$c_1^t(\mathcal{L}, LPM) = \sum_{\sigma \in \mathcal{L}} \sum_{\gamma_i \in \Gamma^{\gamma}_{\sigma,LPM}} \frac{\pi_{time}(\gamma_i(|\gamma_i|)) - \pi_{time}(\gamma_i(1))}{\sum_{\sigma \in \mathcal{L}} \Gamma^{k}_{\sigma,LPM}} \leq max\_time \tag{7.1}$$

$$f_1^e(events(\mathcal{L}), events(\Gamma_{\mathcal{L},LPM}) = \sum_{\sigma \in \mathcal{L}} \sum_{\gamma_i \in \Gamma^{\gamma}_{\mathcal{L},LPM}} \frac{[\pi_{gir\_in}(\gamma_i(1)) - \pi_{gir\_out}(\gamma_i(|\gamma_i|))]^3}{7 - \pi_{gir\_in}(\gamma_i(1))} \tag{7.2}$$

L'Équation 7.1 représente la contrainte de profit définie sur le périmètre trace qui exige que la durée d'exécution d'un fragment rejouable dans un LPM ne doit pas dépasser une durée prédéfinie. Dans notre cas, max_time est égal à 7 jours, limitant le nombre de facteurs impactant la dégradation du niveau de GIR. L'Équation 7.2 représente la fonction de profit définie sur le périmètre event qui évalue le profit en fonction de la dégradation du niveau de GIR. La différence de niveaux de GIR est élevée au cube pour signifier qu'une dégradation de deux niveaux de GIR est plus que deux fois plus grave qu'une dégradation d'un niveau de GIR ; et pour conserver le signe de la différence. La division sert à donner plus d'importance à une dégradation du niveau de GIR d'un patient qui était autonome.

L'extraction des HU-LPMs sur cet event log prend une dizaine de secondes, sur une machine équipée d'un processeur Intel i7 2,4GHz et 16Go de RAM.

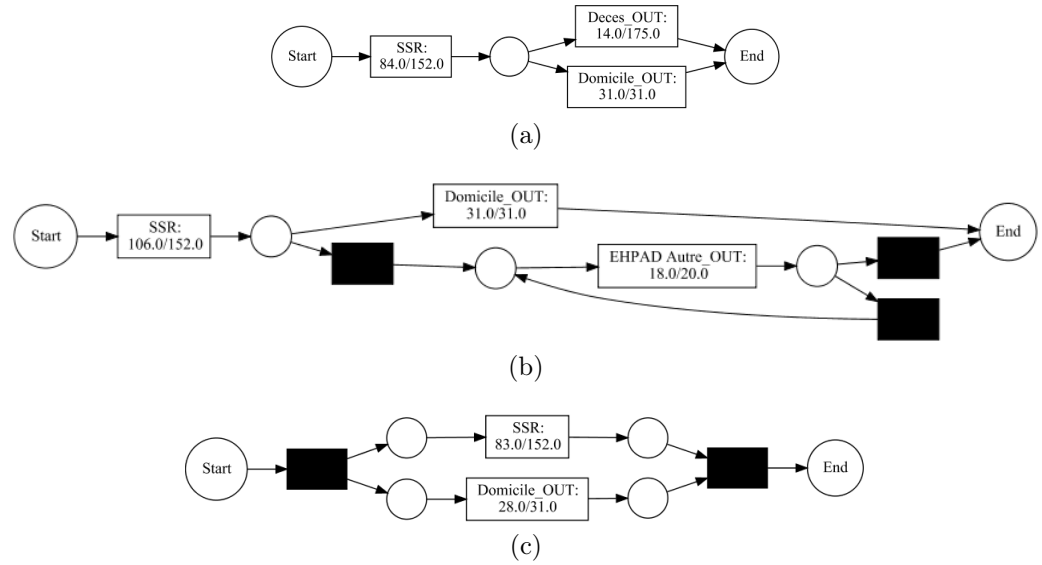


FIGURE 7.4 – Trois HU-LPMs extraits à partir du log CHPCA, en utilisant *la dégradation du niveau de GIR* comme profit.

Nous présentons dans la Figure 7.4 trois HU-LPMs extraits à partir du log CHPCA. Le premier HU-LPM indique qu'un passage en SSR, suivi d'un décès ou d'une sortie à domicile, est accompagné d'une perte d'autonomie. Il est à noter que le niveau de GIR d'une personne décédée est celui constaté les jours qui précèdent le décès. Cela peut s'expliquer par une prise en charge "palliative" de certains patients en SSR, dont le niveau de dépendance peut décroître rapidement. Nous constatons aussi que la baisse d'autonomie des patients qui retournent à leur domicile suit toujours un passage en service de SSR.

Le second HU-LPM insiste sur la réduction du niveau de GIR pour les patients retournant à leur domicile, mais montre aussi que les patients effectuant des séjours répétés au CHPCA, depuis et vers un autre EHPAD que celui de la filière intégrée du CHPCA, ont tendance à voir leur autonomie diminuer.

Enfin, nous présentons le troisième HU-LPM pour compléter les informations portées par le premier HU-LPM, qui montre que tous les patients qui sont retournés à leur domicile et ayant perdu en autonomie étaient passés par le service de SSR. D'après le langage du troisième HU-LPM, nous remarquons que la majorité de la réduction d'autonomie constatée dans les passages en SSR n'était pas précédé d'un passage en MCO. Cela peut signifier que la perte d'autonomie n'est pas due à la prise en charge purement médicale du CHPCA.

7.2 Application des heuristiques pour le problème d'extraction de modèles de processus partiels profitables

Dans cette section, nous présentons différentes applications des heuristiques proposées pour le problème d'extraction de modèles de processus profitables. Dans un premier temps, nous évaluons dans quelle mesure les différentes *configurations* des heuristiques répondent à notre objectif de réduire les temps de calcul, tout en permettant l'extraction de HU-LPMs de bonne qualité. Nous définissons une configuration comme la combinaison d'une heuristique et d'une valeur de k. Dans un second temps, nous explorons les relations entre la performance des configurations en termes de gain de temps de calcul et de qualité des HU-LPMs extraits, et les propriétés du log.

Nous détaillons le contexte des expérimentations et notre méthodologie d'évaluation dans la sous-section 7.2.1 et analysons les résultats dans la sous-section 7.2.2. Nous présentons l'analyse détaillée des relations entre la performance des heuristiques et les caractéristiques du log dans la sous-section 7.2.3.

7.2.1 Contexte d'évaluation

Pour évaluer la performance des différentes configurations heuristiques, nous appliquons chacune d'elles sur une collection d'event logs composée de 3 logs réels issus de la littérature et d'un ensemble de logs générés pour nos expérimentations. Les logs réels sont le *BPI'13 closed problems*, composé de 1 487 traces, du *BPI'13 open problems* composé de 819 traces et d'un log artificiel extrait du "Process Mining book" [van der Aalst, 2016] (Chapitre 1) composé de 6 traces.

Étant donné que nous désirons explorer les relations entre la performance des différentes configurations et les caractéristiques d'un log, nous générons un ensemble de logs aux propriétés diverses. Pour générer les logs, nous utilisons l'outil *PTandLogGenerator* [Jouck and Depaire, 2016], qui permet de générer des process trees aléatoirement, en demandant à l'utilisateur de définir comme paramètres le pourcentage des différents types de noeud-opérateur. Nous générons donc 27 process trees uniques à partir des répartitions présentées dans le Tableau 7.2, avec les types séquence (Séq.), boucle (Bou.), parallélisme (Par.) et choix (Ch.). Comme le temps de calcul nécessaire à l'extraction de LPMs dépend entre autres du nombre d'activités dans le log, il est aussi important de faire varier ce

Log (#)	Séquence	Boucle	Parallélisme	Choix	Log (#)	Séquence	Boucle	Parallélisme	Choix
1	0.4	0.0	0.0	0.6	15	0.5	0.1	0.2	0.2
2	0.4	0.0	0.1	0.5	16	0.5	0.2	0.0	0.3
3	0.4	0.0	0.2	0.4	17	0.5	0.2	0.1	0.2
4	0.4	0.1	0.0	0.5	18	0.5	0.2	0.2	0.1
5	0.4	0.1	0.1	0.4	19	0.5	0.0	0.0	0.5
6	0.4	0.1	0.2	0.3	20	0.5	0.0	0.1	0.4
7	0.4	0.2	0.0	0.4	21	0.5	0.0	0.2	0.3
8	0.4	0.2	0.1	0.3	22	0.6	0.1	0.0	0.3
9	0.4	0.2	0.2	0.2	23	0.6	0.1	0.1	0.2
10	0.5	0.0	0.0	0.5	24	0.6	0.1	0.2	0.1
11	0.5	0.0	0.1	0.4	25	0.6	0.2	0.0	0.2
12	0.5	0.0	0.2	0.3	26	0.6	0.2	0.1	0.1
13	0.5	0.1	0.0	0.4	27	0.6	0.2	0.2	0
14	0.5	0.1	0.1	0.3					

TABLEAU 7.2 – Propriétés des différents logs générés aléatoirement.

nombre entre les différents logs générés. Par conséquent, nous choisissons aléatoirement le nombre d'activités utilisées par chaque process tree généré à partir d'une loi triangulaire ; avec un minimum de 10 activités, un maximum de 30 activités et un mode de 20 activités. Nous simulons 100 traces à partir de chacun des process trees pour générer 27 events logs avec des caractéristiques diverses.

Ces 27 logs générés n'ont pas de *profit* qui pourrait être utilisé pour l'extraction de HU-LPMs. Par conséquent, nous ajoutons artificiellement un attribut *cost* pour chaque log comme base de calcul du profit d'un HU-LPM. Nous ajoutons cet attribut de deux façons. Tout d'abord, nous ajoutons un profit à chaque event de sorte que les différentes valeurs attribuées soient assez homogènes ; i.e., la mesure dans laquelle un event peut contribuer à un HU-LPM varie peu d'un event à un autre. Pour cela, la valeur attribuée est extraite d'une loi Normale avec une moyenne de 10 et un écart-type de 1. Ensuite, nous ajoutons un profit à chaque event de sorte que les différentes valeurs attribuées soient très dispersées ; i.e., une petite partie des events se partage la majorité du profit total du log. Pour cela, la valeur attribuée est extraite d'une loi de Pareto avec une location de 1 et une forme de 1.

Pour chacun des 27 logs, nous générons un log avec des profits issus d'une distribution Normale et un log avec des profit issus d'une distribution de Pareto. Nous avons donc au total 54 logs générés.

A partir de ces 54 logs, la méthode d'évaluation que nous proposons est la suivante. Pour chaque log, nous effectuons une extraction de HU-LPMs "naïve" ; i.e., sans aucune technique d'élagage. De cette manière, nous aurons accès à la liste complète des meilleurs HU-LPMs et à la taille de l'espace de recherche lorsque tous les LPMs sont explorés. Ensuite, nous effectuons une extraction de HU-LPMs pour chacune des 4 heuristiques et {1,2,3} les valeurs de k, pour un total de 12 configurations. Pour rappel, nous considérons les heuristiques *sans mémoire* H_1 et H_2 où H_1 exige que l'expansion d'un LPM soit arrêtée si les k dernières expansions ont généré un LPM avec un profit inférieur ou égal à celui de son parent, et H_2 la version assouplie de H_1 où un LPM a le droit d'avoir un LPM identique à celui de son parent ; et les heuristiques *avec mémoire* H_3 et H_4 où H_3 exige que l'expansion d'un LPM soit arrêtée si les k dernières expansions ont généré un LPM avec un profit inférieur ou égal à celui de son meilleur ancêtre, et H_4 la version assouplie de H_3 où un LPM a le droit d'avoir un LPM identique à celui de son meilleur ancêtre. Nous limitons l'expansion des différents LPMs à 4 expansions successives ; i.e., exp_max=4, pour que

les expérimentations s'effectuent dans un temps raisonnable. Enfin, nous évaluons le gain en termes d'espace de recherche exploré par les différentes configurations. Nous avons volontairement décidé de ne pas évaluer le gain en temps CPU. Certains logs sont très long à analyser, et les temps de calcul nécessaires à l'extraction des HU-LPMs explosent une fois que la consommation en mémoire RAM arrive à saturation. Ces résultats ne sont pas exploitables dans le sens où les ratios calculés ne seraient pas pertinents.

Comme nous l'avons énoncé, l'utilisation d'heuristiques peut empêcher l'extraction de HU-LPM avec un profit élevé, menant à une extraction de moins bonne qualité. Nous appellons une *HU-list* la collection de HU-LPMs extraits par une technique. Soit $\mathcal{S}_a = \langle M_1, M_2, \ldots, M_n \rangle$ la HU-list obtenue par la technique a. Nous appelons *HU-list idéale* la HU-list extraite par une extraction naïve sans élagage. La qualité d'une HU-list dépend de la qualité des HU-LPMs qu'elle contient par rapport à la qualité des HU-LPMs contenus par la HU-list idéale. Pour effectuer cette comparaison, nous utilisons le *normalized Discounted Cumulative Gain* ($nDCG$) [Burges et al., 2005]. Le $nDCG$ est une mesure servant à évaluer un classement, et est une des mesures les plus utilisées dans le domaine de la Recherche d'Information [Tax et al., 2015]. Le *Discounted Cumulative Gain* (DCG) évalue la qualité d'un classement à partir de la pertinence des éléments du classement de telle sorte qu'il donne plus d'importance aux éléments placés en haut du classement. Nous notons rel_i la pertinence d'un élément du classement à la position i. Pour l'évaluation d'une HU-list, nous définissons rel_i comme le profit du HU-LPM à la position i. Nous présentons dans l'Équation 7.4 la définition formelle du DCG pour les p premiers éléments d'un classement.

$$DCG@p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{7.3}$$

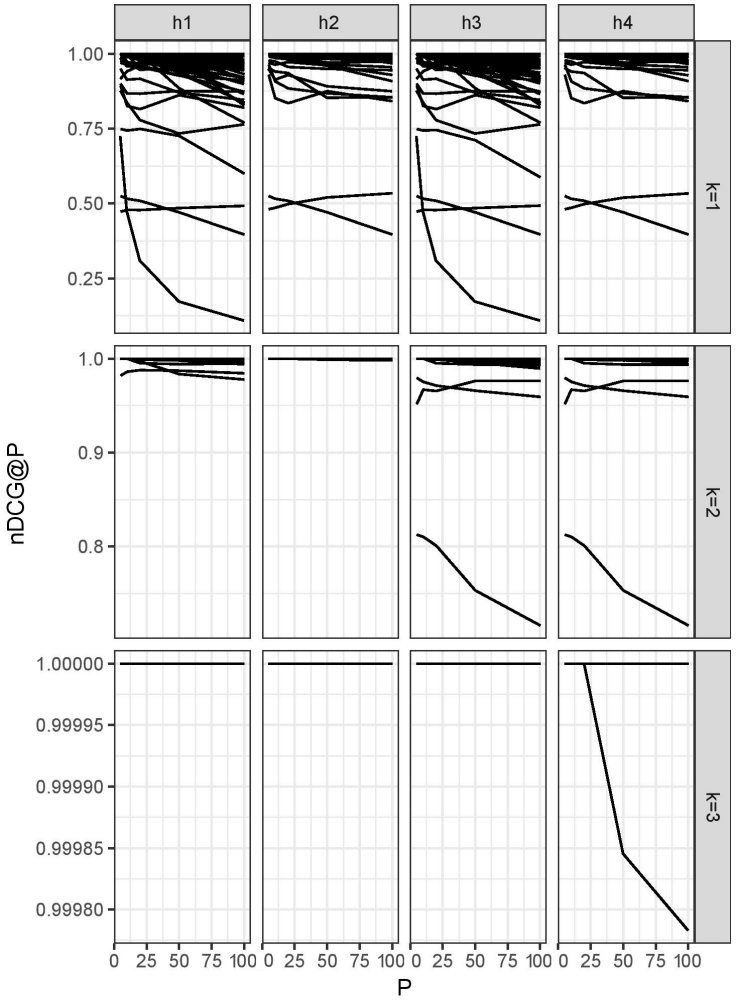
Nous définissons le $IDCG$ comme le DCG obtenu à partir de la HU-list idéale. Nous présentons dans l'Équation 7.4 la formalisation du $nDCG$ à partir du DCG de la HU-list extraite par une configuration et le $IDCG$ de la HU-list idéale :

$$nDCG@p = \frac{DCG@p}{IDCG@p} \tag{7.4}$$

Nous limitons l'évaluation du $nDCG$ aux p premiers HU-LPMs d'une HU-list, noté $nDCG@p$. La borne supérieure de p est le nombre de HU-LPMs dans une HU-list obtenue par une extraction avec élagage avec la méthode a ; $0 < p \leq |\mathcal{S}_a|$.

7.2.2 Analyse des résultats

Nous présentons dans la Figure 7.5a les $nDCG@p$ obtenus pour différentes valeurs de p et pour chaque configuration heuristique. L'axe des abscisses représente la valeur de p utilisée pour calculer le $nDCG@p$, dont la valeur est représentée sur l'axe des ordonnées. Chaque courbe de ce graphe représente un des 54 logs générés artificiellement ou un des trois logs existants. Nous constatons que les 4 heuristiques obtiennent un $nDCG$ proche de 1 sur la plupart des logs. Les heuristiques H_1 et H_3 obtiennent les $nDCG$ les plus bas, surtout avec k=1. Nous présentons dans le Tableau 7.5b, le ratio d'espace de recherche (Ratio Esp. Rech.) exploré ; i.e., quelle part de l'espace de recherche exploré par une extraction "naïve" est explorée par la configuration heuristique pour les 57 logs, en termes de nombre de LPMs générés et évalués.



(a)

		Ratio Esp. Rech.	
H	**k**	**méd.**	**E.A.M**
H_1	**1**	**0.16**	0.10
	2	0.84	0.08
	3	1.00	0.00
H_2	**1**	0.32	0.08
	2	0.88	0.07
	3	1.00	0.00
H_3	**1**	**0.16**	0.10
	2	0.75	0.16
	3	1.00	0.00
H_4	**1**	0.32	0.08
	2	0.75	0.16
	3	0.99	0.02

(b)

FIGURE 7.5 – *(a)* Le $nDCG$ pour chaque configuration heuristique sur les différents logs et *(b)* la réduction de l'espace de recherche en termes de nombre de LPMs explorés lors des différentes expansions.

Pour ne pas être perturbés par des valeurs dispersées, nous utilisons les mesures de *médiane* (méd.) et d'*écart absolu médian* (E.A.M) (plutôt que la moyenne et l'écart-type). Nous constatons que les heuristiques H_1 et H_3 mènent à la plus grosse réduction de l'espace de recherche, avec une médiane à 0,16 indiquant que seulement 16% de l'espace de recherche total est exploré par ces heuristiques avec k=1. Cela correspond à une extraction de HU-LPM potentiellement $\frac{1}{0,16}$=6, 25 fois plus rapide. En utilisant k=3, les heuristiques H_1, H_2 et H_3 explorent l'espace de recherche dans son ensemble, obtenant un $nDCG$ de 1. Pour H_4, il n'y a qu'un seul log pour lequel l'espace de recherche n'est pas exploré entièrement, représenté par la seule courbe avec un $nDCG$ inférieur à 1.

De manière générale, le $nDCG$ pour les différentes configurations et les 57 logs, décroît au fur et à mesure que p augmente. Cela signifie que la majorité des HU-LPMs "ignorés" par l'heuristique sont positionnés dans la partie basse de la HU-list idéale. Par extension, nous pouvons dire que les HU-LPMs les mieux classés sont extraits même en utilisant une heuristique. Les valeurs de l'écart absolu médian que nous présentons dans le Tableau 7.5 montrent aussi que la réduction de l'espace de recherche est sensiblement homogène sur les différents logs pour chaque configuration.

Nous pouvons conclure à partir de ces expérimentations que les heuristiques H_1 et H_3 avec k=1 contribuent à la plus forte réduction de l'espace de recherche, mais ce au détriment de la perte de HU-LPMs de bonne qualité. A l'inverse, k=3 mène à une extraction d'une HU-list identique à la HU-list idéale, mais ne réduit pas l'espace de recherche. Le meilleur compromis est fait par les heuristiques "avec mémoire" avec k=2, étant donné qu'elles réduisent l'espace de recherche d'un quart tout en effectuant une extraction d'une qualité à peine inférieure à l'extraction idéale.

Comme nous venons de le constater, la performance dépend (1) de l'heuristique, (2) de la valeur de k et (3) du log sur lequel la configuration est appliquée. Dans la section suivante, nous essayons d'explorer l'impact des caractéristiques des logs sur la performance des différentes configurations.

7.2.3 Analyse des relations entre la performance des heuristiques et les caractéristiques du log

A partir des performances obtenues dans les expérimentations précédentes, nous voulons voir dans quelle mesure les heuristiques proposées sont impactées par les caractéristiques du log sur lequel elles sont appliquées. Pour cela, nous utilisons des méthodes d'apprentissage supervisé , plus particulièrement de *régression*, pour analyser les interactions entre les caractéristiques du log d'une part, et le ratio d'espace de recherche exploré et la qualité de la HU-list extraite (avec le $nDCG$) d'autre part. Pour rappel, une méthode d'apprentissage supervisé est composée d'une phase d'apprentissage dans laquelle un modèle "apprend" (il "est entraîné"), et une phase de prédiction dans laquelle le modèle tente de prédire la valeur cible d'une nouvelle observation.

Les techniques retenues sont les *arbres de régression*, les *forêts aléatoires* et la *régression linéaire*. Un *arbre de régression* est utilisé pour expliquer et/ou prédire les valeurs prises par une variable dépendante quantitative, en fonction de variables explicatives quantitatives et/ou qualitatives. Une *forêt aléatoire* est un ensemble d'arbres entraînés sur un sous-échantillon du dataset original, constitués aléatoirement. Ces deux méthodes produisent des modèles arborescents, donc assez faciles à comprendre. La *régression linéaire* est une des techniques de régression les plus utilisées, dont l'objectif est de trouver une relation linéaire entre la variable à expliquer et la ou les variable(s) explicative(s). La mesure utilisée pour évaluer la qualité de la régression linéaire, et que nous introduisons plus loin, ne nous permet pas d'utiliser les méthodes "classiques" de régression linéaire ; e.g, la méthode des moindres carrés. Pour pallier ce problème, nous avons eu recours à un réseau de neurones composé d'une seule couche, dont les techniques développées permettent d'optimiser le modèle selon la mesure que nous avons sélectionnée.

Pour essayer d'évaluer l'impact des caractéristiques du log sur les performances des heuristiques, il faut d'abord définir ces caractéristiques. Nous présentons la liste de caractéristiques que nous retenons dans notre analyse :

— **Nombre d'events et d'activités** : représente le nombre d'events et d'activités dans le log. Nous nous attendons à ce que le nombre d'activités impacte négativement la performance des heuristiques, étant donné que le nombre d'expansions possibles dépend en partie du nombre d'activités distinctes présentes dans le log.

— **Nombre de variantes de traces** : représente le nombre de traces distinctes à partir de la projection sur leurs activités. Ce nombre va impacter le nombre de fois

par LPM que la procédure de segmentation va devoir être exécutée [van der Aalst et al., 2012].

— **Nombre moyen d'events et d'activités par trace** : représente la longueur moyenne des traces d'un log et le nombre d'activités distinctes par trace.

— **Taille maximum** : représente la taille de la trace la plus longue. Cette caractéristique est importante dans la mesure où le temps d'exécution de la procédure de segmentation dépend de la longueur de la trace.

De plus, pour répertorier les caractéristiques intéressantes d'un log, nous analysons le *directly-follows graph* (DFG). Le DFG est une structure qu'il est possible de construire depuis un log, et qui est souvent utilisée par les techniques de process mining. Le DFG d'un log $\mathcal{L}$ est un graphe orienté pondéré, dans lequel un sommet représente une activité du log $a \in \Sigma_{\mathcal{L}}$; et dans lequel deux activités $a, b \in \Sigma_{\mathcal{L}}$ sont connectées par un arc si l'activité a est directement suivie de l'activité b dans une des traces de $\mathcal{L}$. Le poids d'un arc représente la fréquence à laquelle cette succession s'est produite dans $\mathcal{L}$. Pour extraire des propriétés intéressantes depuis ce log, nous allons utiliser certaines caractéristiques définies par la *Science des Réseaux* [Lewis, 2011], une discipline qui s'attache à construire des graphes et à extraire des propriétés intéressantes depuis ces graphes. Nous présentons la liste des caractéristiques que nous extrayons depuis le DFG :

— **Diamètre du DFG** : la *distance géodésique* entre deux sommets a et b d'un graphe G est le nombre d'arcs dans le plus court chemin entre a et b, où le plus court chemin est défini en fonction des poids sur les arcs. L'*excentricité* d'un sommet a dans un graphe est définie comme la distance géodésique maximale entre a et n'importe quel autre sommet b de G. Le *diamètre d'un graphe* est l'excentricité maximale parmi tous les sommets du graphe. Le diamètre d'un graphe peut être considéré comme une approximation du degré de connexion des activités dans le log.

— **Centralité d'intermédiarité du DFG** [Freeman, 1977] : Pour V l'ensemble des sommets d'un graphe, la *centralité d'intermédiarité d'un sommet* $v_1 \in V$ est le ratio de plus courts chemins entre deux sommets $v_2, v_3 \in V$ passant pas v_1. Nous élevons cette notion aux graphes en calculant la moyenne des centralités d'intermediarité de chaque sommet du graphe. Une distance courte entre deux sommets a et b dans un DFG signifie que les activités qu'ils représentent semblent n'avoir aucun lien (ils ont une relation faible en termes de fréquence de succession). Un sommet qui a une centralité d'intermédiarité élevée sera sur la périphérie d'un graphe ; i.e., il ne sera pas relié à beaucoup d'autres sommets. La centralité d'intermédiarité d'un DFG peut donc être perçue comme la *séquentialité* d'un log ; i.e., elle sera élevée si le processus est séquentiel, et sera basse si le processus est aléatoire ou comporte beaucoup de situations de parallélisme.

— **Densité du DFG** : représente le ratio du nombre d'arcs dans un graphe sur le nombre d'arcs possibles dans ce même graphe ($|\Sigma_L|^2$ pour un DFG).

— **PageRank du DFG** [Brin and Page, 1998] : *PageRank* est un des algorithmes les plus connus pour mesurer l'importance relative des sommets d'un graphe. C'est un algorithme itératif qui, à chaque itération, met à jour l'importance d'un sommet en prenant en compte l'importance des sommets qui sont liés à lui. [Xing and Ghorbani, 2004] ont étendu cet algorithme aux graphes avec arcs pondérés. Nous appliquons cette extension du PageRank aux DFG. De cette façon, les activités qui sont fréquentes dans le log et qui sont fréquemment suivies et précédées par

d'autres activités fréquentes auront un PageRank élevé. Plusieurs sommets avec un PageRank élevé peuvent être interprétés comme le fait qu'il y a beaucoup de LPM fréquents dans le log. A partir du PageRank pour les différents sommets du DFG, nous retenons comme caractéristiques du log le PageRank maximum et la variance.

Les différentes caractéristiques du DFG représentent la partie control-flow de l'event log. Dans nos travaux, nous nous intéressons plus particulièrement au profit des différents events du log, ce qui n'est pas représenté dans le DFG. Nous introduisons une variante du DFG, que nous appelons *utility directly-follows graph* (uDFG). Dans un uDFG, le poids d'un arc allant d'un sommet a vers un sommet b dépend du profit des events b qui succèdent aux events a. Chaque caractéristique définie pour le DFG vaut aussi pour le uDFG. Nous définissons une caractéristique additionnelle pour mesurer le degré d'inégalité des profits entre les events d'une trace. La raison derrière la définition de cette caractéristique est que si les profits sont répartis uniformément entre les events, il y a de fortes chances qu'il y ait un nombre de HU-LPMs plus important, et donc un espace de recherche plus grand. Pour obtenir cette caractéristique, nous utilisons l'*index de Gini* [Gini, 1997], une des mesures d'inégalité les plus utilisées.

L'objectif de notre analyse est de *prédire* la valeur d'une variable. Pour évaluer la précision de nos modèles, nous utilisons la méthode d'estimation *leave-one-out cross-validation* (LOOCV). A partir de l'ensemble de log à notre disposition, la méthode LOOCV effectue la phase d'apprentissage sur tous les logs sauf un ; et prédit puis évalue la qualité de la prédiction sur le log restant. Cette procédure est ensuite répétée pour chaque log à prédire. La précision du modèle est la moyenne de la précision mesurée à chaque itération. Pour évaluer la précision d'une prédiction, nous utilisons la mesure *Mean Absolute Percentage Error* (MAPE). Les mesures plus classiques telles que la *Mean Absolute Error* (MAE) ou la *Mean Squared Error* (MSE) ont l'inconvénient d'être influencées par l'erreur sur la valeur cible la plus élevée. Comme nous avons des valeurs à prédire à la fois très grandes et très petites, nous ne pouvons pas utiliser ces mesures ; e.g., prédire une valeur de 15 au lieu de 10 serait une erreur 10 fois plus grosse que prédire une valeur de 1,5 au lieu de 1. Dans notre cas, nous voulons que l'erreur sur ces deux prédictions soit équivalente ; i.e. de 33%, d'où notre choix d'utiliser la MAPE. La mesure MAPE est définie comme suit :

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} |\frac{A_i - P_i}{A_i}| \tag{7.5}$$

avec n le nombre de prédictions, A_i la valeur réelle de la variable à prédire et P_i la valeur de la prédiction faite par le modèle. Cette mesure indique, en pourcentage, à quel point la prédiction est éloignée de la valeur réelle à prédire, sans faire la distinction entre une prédiction au-dessus ou au-dessous de la valeur réelle.

Nous présentons dans la Figure 7.6 l'arbre de régression obtenu pour l'exploration de la relation entre la performance des heuristiques en termes de ratio d'espace de recherche et les caractéristiques du log. Un arbre de régression est un outil d'aide à la décision représentant un ensemble de choix sous la forme graphique d'un arbre. Les différentes décisions possibles sont situées au niveau des nœuds de l'arbre. Chaque nœud décrit la distribution de la variable à prédire. Dans le cas du premier nœud de l'arbre présenté dans la Figure 7.6, nous constatons qu'il y a 408 observations (*samples*) dans le log. A ce stade là, la variable à prédire prend la valeur (*value*) 0,527. Les enfants d'un nœud représentent un "split" binaire d'une caractéristique du log. Par exemple, pour le premier nœud de l'arbre, si $k <= 1,5$, nous nous dirigeons vers la partie haute de l'arbre, sinon

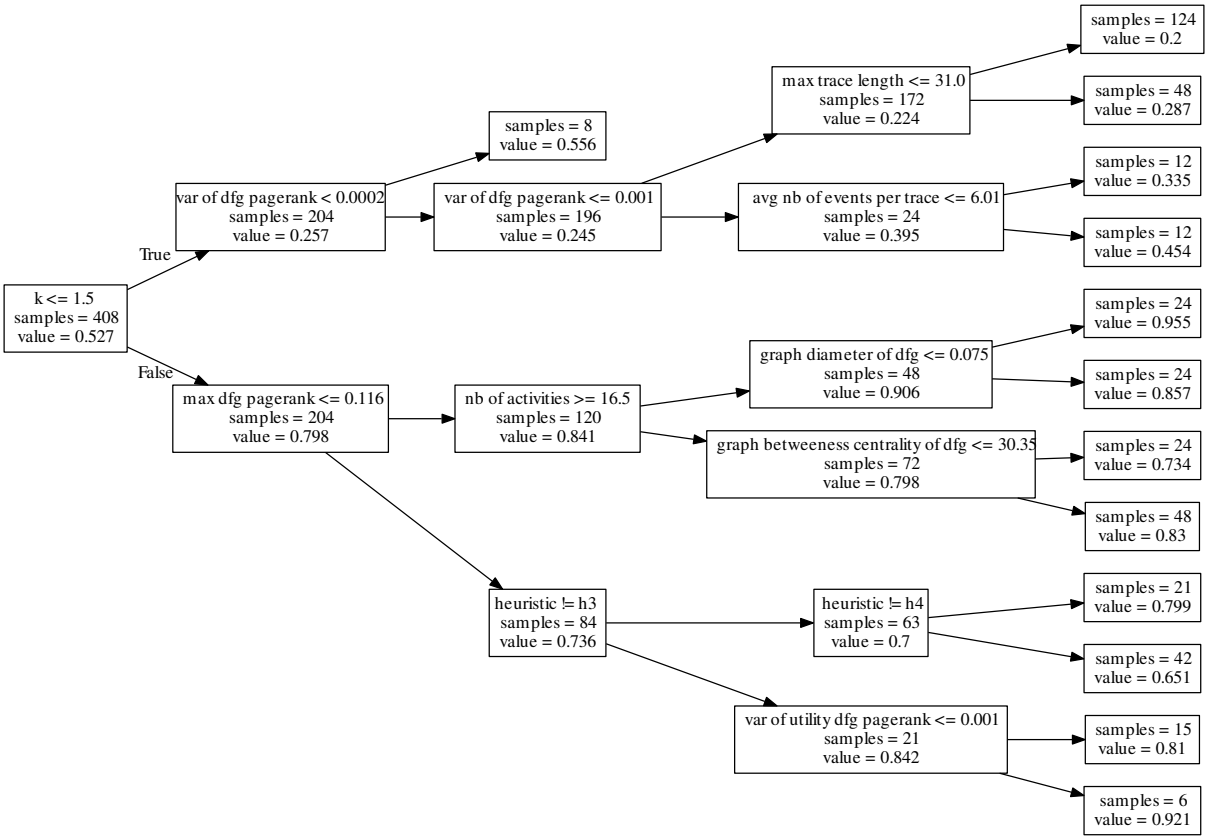


FIGURE 7.6 – Arbre de régression montrant la relation entre les caractéristiques du log et le ratio d'espace de recherche.

vers la partie basse, et ainsi de suite pour chaque caractéristique utilisée dans l'arbre. La valeur indiquée dans la variable *samples* indique le nombre d'observations qui respectent toutes les décisions prises jusqu'à ce nœud de l'arbre. L'objectif est d'affiner la valeur à prédire au fur et à mesure que de nouvelles caractéristiques sont prises en compte.

Nous avons filtré les valeurs pour k=3 puisque nous avons vu dans la sous-section 7.2.2 que le ratio d'espace de recherche est de 1. Donc il n'est pas utile de les considérer dans l'analyse. Le premier split sépare l'arbre selon si k=1 ou k=2. Cela signifie que la valeur de k est un facteur prépondérant dans la performance d'une heuristique et nous constatons que la valeur de prédiction est bien plus élevée pour k=2. Cependant, pour k=1, nous remarquons que la prédiction en termes de ratio d'espace de recherche dépend encore de caractéristiques du log. La variance du PageRank du DFG semble avoir une importance particulière ; à la fois la variance la plus haute et la variance la plus basse impliquent des ratios d'espace de recherche plus élevés, alors que des variances intermédiaires semblent impliquer des ratios plus bas. Pour k=2, nous constatons que les heuristiques H_1 et H_2 ont des ratios plus élevés comparé à H_4 ; ce qui est en accord avec les résultats obtenus dans la sous-section précédente. L'arbre montre aussi qu'un nombre important d'activités dans le log (>16,5) implique un ratio d'espace de recherche plus grand.

Nous présentons dans la Figure 7.7 l'arbre de régression construit pour l'exploration de la relation entre la performance des heuristiques en termes de $nDCG$@100 et les caractéristiques du log. La plupart des attributs utilisés dans cet arbre sont des caractéristiques du log ; e.g., le nombre d'activités, le nombre d'events ou le nombre moyen d'events par trace. Cela signifie que, indépendamment de la configuration heuristique utilisée, la structure du

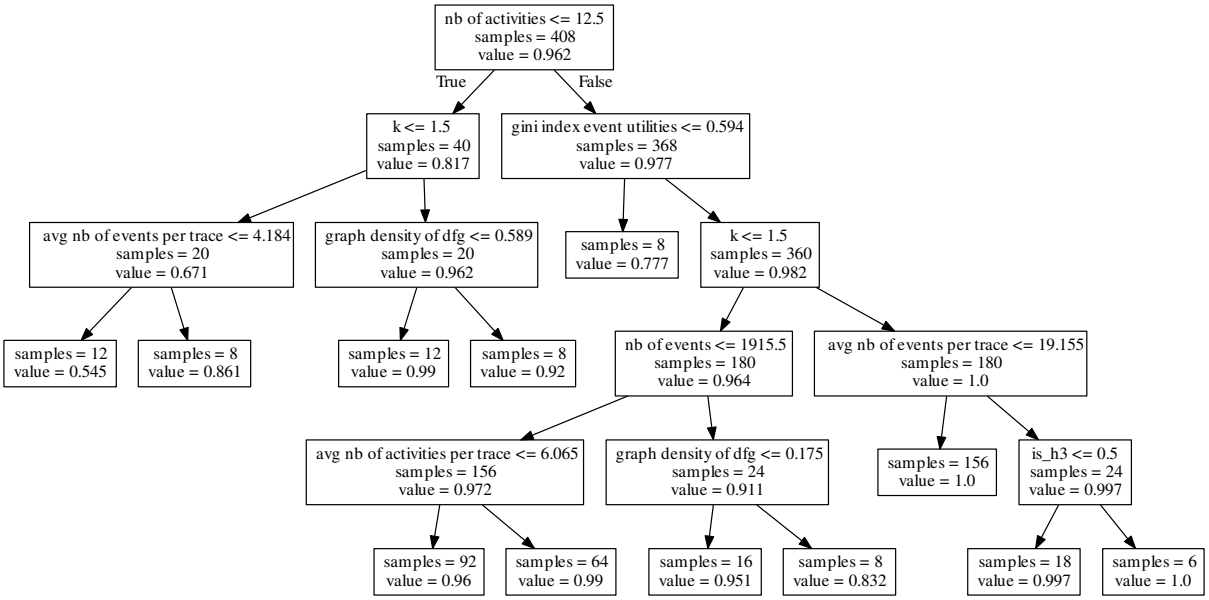


FIGURE 7.7 – Arbre de régression montrant la relation entre les caractéristiques du log et la qualité de la HU-list extraite ($nDCG$@100).

log a un impact sur la qualité de la HU-list extraite plus important que l'heuristique choisie. Nous constatons que k=2 permet d'extraire une HU-list de meilleure qualité que k=1. Avec surprise, nous remarquons que le $nDCG$@100 est meilleur pour les log contenant au moins 13 activités. Les logs pour lequel l'index de Gini est bas ; i.e., où le profit est uniformément réparti entre les events, ont un $nDCG$@100 plus bas. Nous pouvons l'expliquer par le fait qu'avec une répartition uniforme des profits, il y a plus de HU-LPMs à extraire. Cela mène à un espace de recherche total plus grand, et par extension à une phase d'élagage plus complexe, lors de laquelle il est beaucoup plus facile de "manquer" un HU-LPM de bonne qualité.

Technique	Ratio d'espace de recherche		nDCG@100	
	moyenne	écart-type	moyenne	écart-type
Moyenne globale	825.83	927.21	4.32	7.84
Moyenne heuristique	382.75	463.84	3.97	7.62
Arbres de régression	121.63	159.41	3.57	8.20
Forêts aléatoires	177.20	185.24	3.74	7.42
Régression linéaire	**86.60**	21.81	**3.15**	7.12

TABLEAU 7.3 – Moyenne et écart-type de la MAPE pour différentes techniques de prédiction.

Nous présentons dans le Tableau 7.3 les résultats en termes de MAPE des différentes techniques de prédiction pour le ratio d'espace de recherche et le $nDCG$@100. Nous utilisons deux bases de comparaison : la *moyenne globale* et la *moyenne heuristique*. La moyenne globale représente la moyenne de ratio et de $nDCG$@100 pour toutes les configurations sur tous les events logs. La moyenne heuristique représente la moyenne de ratio et de $nDCG$@100 des observations ayant la même configuration que celle prédite.

La moyenne heuristique a de meilleurs résultats que la moyenne globale, ce qui signifie

que les performances des heuristiques dépendent de l'heuristique et de la valeur de k choisies. Les arbres de régressions présentés dans les Figures 7.6 et 7.7 permettent des prédictions plus précises que la moyenne heuristique. Cela montre qu'il y a un intérêt à explorer les caractéristiques des logs pour prédire le ratio d'espace de recherche et le $nDCG$@100. Cependant, la régression linéaire est de loin la méthode qui donne les meilleurs résultats pour chaque variable à prédire.

Il est à noter que les valeurs de la MAPE pour la prédiction du ratio d'espace de recherche sont très élevées, alors que les valeurs de la MAPE pour la prédiction du $nDCG$@100 sont très basses. Cette différence peut s'expliquer par le fait que les valeurs prises pour le ratio d'espace de recherche sont sur un grand intervalle, avec certaines valeurs proche de 0 et d'autres proches de 1, alors que les valeurs prises pour le $nDCG$@100 sont concentrées autour de 1. Cela rend la tâche de prédire le ratio d'espace de recherche beaucoup plus difficile que celle de prédire la valeur du $nDCG$@100.

Toutefois, le fait que les erreurs de prédiction en utilisant les caractéristiques du log soient largement inférieures aux bases de comparaison prouve que le ratio d'espace de recherche peut être prédit avec une certaine précision lorsque les caractéristiques du log sont prises en compte.

Conclusion

Dans ce chapitre, nous avons dans un premier temps testé la capacité des contraintes et fonctions de profit à permettre l'extraction de modèles de processus partiels profitables, les HU-LPMs. Ces expérimentations ont été menées sur quatre logs issus de domaines d'application différents. Nous avons par conséquent démontré que notre approche était générique et capable d'extraire des informations intéressantes à partir de processus peu structurés.

Dans un second temps, nous avons évalué les performances des heuristiques proposées. Nous avons démontré qu'en plus de réduire efficacement les temps de calcul, les heuristiques permettent l'extraction d'une collection de HU-LPMs presque optimale. La perte d'informations due aux HU-LPMs manquants est nettement compensée par le gain de temps nécessaire à l'extraction.

Enfin, nous avons constaté que les heuristiques n'avaient pas les même résultats en termes de temps de calcul et de qualité de l'extraction en fonction du log utilisé. A l'aide de modèles d'apprentissage supervisé, nous avons tenté de détecter des relations entre différentes caractéristiques des logs et les performances des heuristiques. Bien que les méthodes proposées ont des prédictions éloignées de la réalité, nous avons démontré qu'elles étaient bien meilleures que des mesures statistiques classiques. Cela signifie que les performances des heuristiques sont bien impactées par la structure des logs sur lesquels elles sont appliquées.

Conclusion de Partie

Les approches proposées dans la première partie étaient limitées lorsque le processus analysé est peu structuré. L'objectif de cette seconde partie était de proposer une alternative à ces approches.

Dans le chapitre 5, nous avons détaillé deux alternatives à la modélisation de processus avec des modèles globaux. Dans un premier temps, nous avons présenté les modèles locaux qui représentent des fragments de processus. Dans un second temps, nous avons présenté les modèles déclaratifs, qui fournissent un ensemble de règles que les différentes exécutions du processus doivent respecter. Dans une dernière section, nous avons introduit la technique de LPM Discovery, une technique qui génère une collection de modèles locaux et à partir de laquelle nous nous inspirons pour la suite de nos travaux.

Dans le chapitre 6, nous étendons le problème d'extraction de modèles de processus partiels pour prendre en compte l'importance relative des différents éléments d'un event log. Dans la première section nous avons donc formalisé le problème d'extraction de modèles de processus partiels profitables à l'aide d'un ensemble de contraintes et fonctions de profit. Dans une seconde section, nous avons proposé des méthodes heuristiques pour réduire les temps de calculs nécessaires à l'extraction.

Dans le chapitre 7, nous avons présenté quatre études de cas. Les jeux de données utilisés sont issus de domaines différents pour tester la généricité de nos approches. Nous avons montré que les approches proposées permettent d'extraire des fragments de comportements plus intéressants qu'avec une approche basée sur la fréquence, quels que soient le processus et le domaine d'application utilisés.

Nous avons ensuite évalué à l'aide de techniques d'apprentissage supervisé l'impact des caractéristiques d'un log sur l'efficacité des heuristiques.

Conclusion Générale

La France est confrontée à un vieillissement de la population. En plus de l'augmentation du nombre de personnes âgées, nous constatons que les pathologies développées deviennent de plus en plus chroniques. Par conséquent, la demande en services de soins augmentent. Dans un contexte de réduction des moyens, le système de santé se doit de mieux maîtriser ses processus de santé. Le projet SaAge, dans lequel nos travaux s'inscrivent, a vu le jour dans l'optique de mieux comprendre le parcours de santé de la personne âgée en Auvergne.

Dans ce manuscrit, nous nous sommes intéressés à deux problématiques. La première consiste à détecter des relations de dépendance entre les activités d'un processus. Plus particulièrement, nous voulons savoir si les choix qui sont faits dans un processus, et l'ordre dans lequel ils sont faits, impactent la suite du processus. La seconde problématique consiste à proposer une méthode d'évaluation de ces dépendances en fonction de l'importance relative des activités concernées. Selon le point de vue métier avec lequel nous analysons un processus, certaines caractéristiques seront plus intéressantes que d'autres.

Pour répondre à ces problématiques, nous utilisons dans nos travaux le process mining pour tirer profit des données disponibles dans le domaine de la santé. Plusieurs approches ont été envisagées. Dans une première partie, nous avons proposé la méthodologie *KITE*, dont l'objectif est de s'appuyer sur le pouvoir d'expression des modèles de processus pour extraire des connaissances intéressantes à partir des données. Dans l'optique d'extraire des relations de dépendance entre activités, nous avons proposé *PRISM*, une instanciation de la méthodologie KITE. A partir d'un process tree, PRISM effectue dans un premier temps un cycle de sélection/réduction pour réduire l'espace de recherche à explorer. Dans un second temps, il s'appuie sur *TWINCLE*, un algorithme d'extraction de règles séquentielles à bas coût que nous avons développé.

Nous avons montré que les approches proposées sont efficaces et permettent d'extraire des relations de dépendances intéressantes. Cependant nous avons remarqué que nos approches sont moins efficaces lorsque nous analysons un processus peu structuré.

Pour palier ces problèmes, nous avons présenté dans une seconde partie une alternative aux approches proposées dans la première partie. Afin d'extraire des informations intéressantes à partir de processus peu structurés, nous avons formalisé le problème d'extraction de modèles de processus partiels profitables. Ce problème consiste à extraire, à l'aide de contraintes et fonctions de profit, une collection de HU-LPMs ; i.e., un ensemble de fragments de comportements intéressants selon un point de vue métier prédéfini. Ce problème étant un problème combinatoire, nous avons proposé plusieurs approches heuristiques afin

de réduire les temps de calcul nécessaires à l'extraction des HU-LPMs.

Nous avons montré que ces nouvelles approches permettent d'extraire des informations très intéressantes à partir de tout type de processus. De plus, nous avons pu montrer l'aspect générique de nos méthodes en les appliquant à divers domaines. Les résultats que nous obtenons prouvent que nos méthodes sont capables de réduire les temps de calculs tout en effectuant une extraction de grande qualité. De plus, nous avons analysé à l'aide de modèles prédictifs dans quelle mesure certaines caractéristiques d'un log influent sur la performance des méthodes proposées.

Perspectives

Les résultats que nous avons obtenus sont très prometteurs pour nos travaux futurs et nous envisageons différentes perspectives.

Les modèles prédictifs développés dans la seconde partie de nos travaux valident le postulat selon lequel certaines caractéristiques du log influent sur les performances des heuristiques utilisées. Cependant en l'état, ces résultats ne permettent à un utilisateur que de choisir une configuration heuristique pour effectuer l'extraction de HU-LPMs. Une perspective intéressante serait de pousser l'analyse des relations entre caractéristiques des logs et la performance des heuristiques plus loin. Une nouvelle approche serait d'utiliser les différentes caractéristiques *durant* la procédure d'expansion des LPMs, pour décider en temps réel si un LPM doit être étendu ou non. Dans le cas ou un LPM est autorisé à être étendu, il pourrait être envisageable de décider comment l'étendre ; i.e., quel opérateur et quelle activité choisir pour maximiser le profit du nouveau LPM. Une nouvelle heuristique verrait le jour.

Une autre perspective à envisager consiste à améliorer l'extraction des HU-LPMs. Avec les techniques proposées un nombre encore trop important de HU-LPMs est extrait. De plus, il arrive que plusieurs HU-LPMs soient similaires en termes de comportements, et véhiculent donc plus ou moins les même informations. Parmi les extensions apportées par les travaux en extraction de motifs, les motifs *fermés* et *maximaux* permettent de réduire le nombre de motifs extraits tout en permettant l'extraction de motifs de meilleure qualité. En nous inspirant de ces travaux, nous pourrions envisager de définir de nouveaux type de modèles de processus partiels profitables, les HU-LPMs fermés et les HU-LPMs maximaux, pour extraire des modèles encore plus intéressants.

Enfin, une dernière perspective pourrait consister à définir des contraintes de répartition entre les différents "secteurs" du processus. Selon les contraintes et fonctions de profit définies, il se peut que tous les HU-LPMs extraits se concentrent sur des comportements issus de la même partie du processus, laissant le reste du processus peu voire pas analysé du tout. Pour avoir un aperçu global du processus à partir d'un ensemble de fragments, il faudrait pouvoir définir différentes parties dans un processus ainsi qu'un minimum de HU-LPMs à allouer à la modélisation de chacune de ces parties.

Bibliographie

[Aggarwal and Han, 2014] Aggarwal, C. C. and Han, J. (2014). *Frequent Pattern Mining*. Springer International Publishing.

[Agrawal et al., 1993] Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216.

[Agrawal and Srikant, 1994] Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. *Proceedings of the 20th Conference on Very Large Data Bases*, pages 487–499.

[Agrawal and Srikant, 1995] Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering*, pages 3–14.

[Ahmed et al., 2010] Ahmed, C. F., Tanbeer, S. K., and Jeong, B.-S. (2010). A novel approach for mining high-utility sequential patterns in sequence databases. *Electronics and Telecommunications Research Institute Journal*, pages 676—-686.

[Ahmed et al., 2014] Ahmed, C. F., Tanbeer, S. K., and Jeong, B.-S. (2014). A framework for mining high utility web access sequences. *International Electronics and Telecommunications Engineering Technical Review*, pages 3–16.

[Ahmed et al., 2009] Ahmed, C. F., Tanbeer, S. K., Jeong, B.-S., and Lee, Y.-K. (2009). Efficient tree structures for high utility pattern mining in incremental databases. *IEEE Transactions on Knowledge and Data Engineering*, pages 1708–1721.

[Alberti et al., 2008] Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., and Torroni, P. (2008). Verifiable agent interaction in abductive logic programming : the sciff framework. *ACM Transactions on Computational Logic*, 9(4) :29.

[Angles, 2015] Angles, R. (2015). *V-BPMI : une approche pour la conception, l'organisation et l'adaptation de processus flexibles : Cas d'application : la sécurisation du circuit du médicament au sein de l'AP-HM*. PhD thesis, Université Aix-Marseille.

[Aseervatham et al., 2006] Aseervatham, S., Osmani, A., and Emmanuel, V. (2006). bitspade : A lattice-based sequential pattern mining algorithm using bitmap representation. *6th International Conference On Data Mining*, pages 792–797.

[Augusto et al., 2017] Augusto, A., Conforti, R., Dumas, M., and La Rosa, M. (2017). Split miner : Fast automated discovery of simple and accurate bpmn process models. In *International Conference on Data Mining*, page To appear. IEEE.

[Ayres et al., 2002] Ayres, J., Flannick, J., Gehrke, J., and Yiu, T. (2002). Sequential pattern mining using a bitmap representation. In *Proceedings of the Eighth ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 429–435.

[Barber et al., 2017] Barber, R. M., Fullman, N., and Sorensen, R. J. D. (2017). Healthcare access and quality index based on mortality from causes amenable to personal health care in 195 countries and territories, 1990–2015 : a novel analysis from the global burden of disease study 2015. *The Lancet*.

[Benabdejlil et al., 2014] Benabdejlil, H., Alix, T., and Vallespir, B. (2014). Cartographie des acteurs et des processus de soins support à de nouveaux services de santé à forte valeur ajoutée. In *Journées du Réseau Associatif pour l' Amélioration de la Qualité en Santé*.

[Bergenthum et al., 2007] Bergenthum, R., Desel, J., Lorenz, R., and Mauser, S. (2007). Process mining based on regions of languages. In *International Conference on Business Process Management*, pages 375–383. Springer.

[Bloch and Hénaut, 2014] Bloch, M.-A. and Hénaut, L. (2014). *Coordination et parcours. La dynamique du monde sanitaire, social et médico-social*. Santé Social. Dunod.

[Blum et al., 2008] Blum, T., Padoy, N., Feußner, H., and Navab, N. (2008). Workflow mining for visualization and analysis of surgeries. *International Journal of Computer Assisted Radiology and Surgery*, pages 379–386.

[Bose et al., 2011] Bose, R. P. J. C., van der Aalst, W. M. P., Žliobaite, I., and Pechenizkiy, M. (2011). Handling concept drift in process mining. In *Proceedings of the 23rd International Conference on Advanced Information Systems Engineering*, pages 391–405.

[Brin and Page, 1998] Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, pages 107–117.

[Buijs, 2014] Buijs, J. (2014). *Flexible Evolutionary Algorithms for Mining Structures Process Models*. PhD thesis, Technical University of Eindhoven.

[Buijs et al., 2011] Buijs, J. C. A. M., van Dongen, B. F., and van der Aalst, W. M. P. (2011). Towards cross-organizational process mining in collections of process models and their executions. In *Business Process Management Workshops (2)*, pages 2–13.

[Burattin et al., 2016] Burattin, A., Maggi, F. M., and Sperduti, A. (2016). Conformance checking based on multi-perspective declarative process models. *Expert Systems with Applications*, pages 194–211.

[Burges et al., 2005] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of the 22Nd International Conference on Machine Learning*, pages 89–96.

[Cardoen et al., 2010] Cardoen, B., Demeulemeester, E., and Beliën, J. (2010). Operating room planning and scheduling : A literature review. *European Journal of Operational Research*, pages 921–932.

[Cardoso and van der Aalst, 2009] Cardoso, J. S. and van der Aalst, W. M. P. (2009). Path mining and process mining for workflow management systems. In *Encyclopedia of Data Warehousing and Mining*, pages 1489–1496. IGI Global.

[Chabrol et al., 2016] Chabrol, M., Dalmas, B., Norre, S., and Rodier, S. (2016). A process tree-based algorithm for the detection of implicit dependencies. In *IEEE* 10^{th} *International Conference on Research Challenges in Information Science*, pages 1–11.

[Chan et al., 2003] Chan, R., Yang, Q., and Shen, Y.-D. (2003). Mining high-utility itemsets. *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 19–26.

[Chang, 2011] Chang, J. H. (2011). Mining weighted sequential patterns in a sequence database with a time-interval weight. *Knowledge-Based Systems*, pages 1–9.

[Chang and Lee, 2003] Chang, J. H. and Lee, W. S. (2003). Finding recent frequent itemsets adaptively over online data streams. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining*, pages 487–492.

[Ciccio and Mecella, 2015] Ciccio, C. D. and Mecella, M. (2015). On the discovery of declarative control flows for artful processes. *ACM Transactions on Management Information Systems*, 5(4) :24.

[Dalmas et al., 2017a] Dalmas, B., Fournier-Viger, P., and Norre, S. (2017a). Twincle : A constrained sequential rule mining algorithm for event logs. *Procedia Computer Science*, pages 205–214.

[Dalmas et al., 2017b] Dalmas, B., Norre, S., Chabrol, M., and Rodier, S. (2017b). Kite : a process-based mining methodology applied to the extraction of routing rules. In 20^{th} *International Federation of Automamatic Control World Congress*.

[Dalmas et al., 2017c] Dalmas, B., Tax, N., and Norre, S. (2017c). Heuristics for high-utility local process model mining. In 3^{rd} *Workshop on Algorithms & Theories for the Analysis of Event Data*.

[de Leoni et al., 2016] de Leoni, M., van der Aalst, W. M. P., and Dees, M. (2016). A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. *Information Systems*, pages 235–257.

[de Medeiros Ana K. et al., 2007] de Medeiros Ana K., A., Weijters, T. A., and van der Aalst, W. M. P. (2007). Genetic process mining : an experimental evaluation. *Data Mining and Knowledge Discovery*, pages 245–304.

[Di Ciccio et al., 2016] Di Ciccio, C., Maggi, F. M., and Mendling, J. (2016). Efficient discovery of target-branched declare constraints. *Information Systems*, pages 258–283.

[Diamantini et al., 2016a] Diamantini, C., Genga, L., and Potena, D. (2016a). Behavioral process mining for unstructured processes. *Journal of Intelligent Information Systems*, 47(1) :5–32.

[Diamantini et al., 2013] Diamantini, C., Genga, L., Potena, D., and Storti, E. (2013). Pattern discovery from innovation processes. In *International Conference on Collaboration Technologies and Systems*, pages 457–464. IEEE.

[Diamantini et al., 2016b] Diamantini, C., Genga, L., Potena, D., and van der Aalst, W. M. P. (2016b). Building instance graphs for highly variable processes. *Expert Systems with Applications*, pages 101–118.

[Erwin et al., 2008] Erwin, A., Gopalan, R. P., and Achuthan, N. R. (2008). Efficient mining of high utility itemsets from large datasets. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 554–561.

[Faure et al., 2009] Faure, R., Lemaire, B., and Picouleau, C. (2009). *Précis de recherche opérationnelle - 6e éd. : Méthodes et exercices d'application*. Mathématiques. Dunod.

[Fournier-Viger et al., 2012a] Fournier-Viger, P., Gueniche, T., and Tseng, V. S. (2012a). Using partially-ordered sequential rules to generate more accurate sequence prediction. In *International Conference on Advanced Data Mining and Applications*, pages 431–442.

[Fournier-Viger and Tseng, 2013] Fournier-Viger, P. and Tseng, V. S. (2013). Tns : Mining top-k non-redundant sequential rules. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 164–166.

[Fournier-Viger et al., 2012b] Fournier-Viger, P., Wu, C.-W., and Tseng, V. S. (2012b). Mining top-k association rules. In *Proceedings of the 25th Canadian Conference on Advances in Artificial Intelligence*, pages 61–73.

[Fournier-Viger et al., 2013] Fournier-Viger, P., Wu, C.-W., and Tseng, V. S. (2013). Mining maximal sequential patterns without candidate maintenance. In *International Conference on Data Mining and Applications*, pages 169–180.

[Fournier-Viger et al., 2015] Fournier-Viger, P., Wu, C.-W., Tseng, V. S., Cao, L., and Nkambou, R. (2015). Mining partially-ordered sequential rules common to multiple sequences. *IEEE Transactions on Knowledge and Data Engineering*, pages 2203–2216.

[Fournier-Viger et al., 2012c] Fournier-Viger, P., Wu, C.-W., Tseng, V. S., and Nkambou, R. (2012c). Mining sequential rules common to several sequences with the window size constraint. In *Canadian Conference on Artificial Intelligence*, pages 299–304.

[Fournier-Viger et al., 2014] Fournier-Viger, P., Wu, C.-W., Zida, S., and Tseng, V. S. (2014). Fhm : Faster high-utility itemset mining using estimated utility co-occurrence pruning. In *International Symposium on Methodologies for Intelligent Systems*, pages 83–92.

[Frawley et al., 1992] Frawley, W. J., Piatetsky-Shapiro, G., and Matheus, C. J. (1992). Knowledge discovery in databases : An overview. *Artificial Intelligence Magazine*, pages 57–70.

[Freeman, 1977] Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41.

[Gini, 1997] Gini, C. (1997). Concentration and dependency ratios. *Rivista di politica economica*, pages 769–792.

[Günther et al., 2008] Günther, C. W., Rozinat, A., van der Aalst, W. M., and van Uden, K. (2008). Monitoring deployed application usage with process mining. *BPM Center Report*.

[Günther and van der Aalst, 2007] Günther, C. W. and van der Aalst, W. M. (2007). Fuzzy mining adaptive process simplification based on multi-perspective metrics. *Business Process Management*, pages 328–343.

[Gourgand and Kellert, 1991] Gourgand, M. and Kellert, P. (1991). Conception d'un environnement de modélisation des systèmes de production. *3e Congrès International de Génie Industriel*.

[Grahne and Jianfei, 2003] Grahne, G. and Jianfei, Z. (2003). High performance mining of maximal frequent itemsets. In *Proceedings of the 6th International Conference on Innovations in Information Technology*, pages 11–15.

[Grahne and Jianfei, 2005] Grahne, G. and Jianfei, Z. (2005). Fast algorithms for frequent mining using fp-trees. In *IEEE Transactions on Knowledge and Data Engineering*, pages 1347–1362.

[Grissa, 2013] Grissa, D. (2013). *Etude comportementale des mesures d'intérêt d'extraction de connaissance*. PhD thesis, Université Blaise Pascal.

[Han et al., 2004] Han, J., Pei, J., Yiwen, Y., and Mao, R. (2004). Mining frequent patterns without candidate generation. In *Data Mining and Knwoledge Discovery*, pages 53–87.

[Harms et al., 2002] Harms, S. K., Deogun, J., and Tadesse, T. (2002). Discovering sequential association rules with constraints and time lags in multiple sequences. In *International symposium in Methodologies for Intelligent Systems*, pages 432–441.

[He et al., 2016] He, W., Goodkind, D., and Kowal, P. (2016). *An Aging World : 2015, International Population Report*. National Institute on Aging (NIA).

[Hofstede et al., 2009] Hofstede, A., van der Aalst, W., Adams, M., and Russell, N. (2009). *Modern Business Process Automation : YAWL and Its Support Environment*. Springer.

[Huff and Geis, 1954] Huff, D. and Geis, I. (1954). *How To Lie With Statistics*. W. W. Norton and Company.

[Jouck and Depaire, 2016] Jouck, T. and Depaire, B. (2016). Ptandloggenerator : A generator for artificial event data. In *International Conference on Business Process Management*.

[Katz, 1983] Katz, S. (1983). Assessing self-maintenance : activities of daily living, mobility, and instrumental activities of daily living. *Journal of the American Geriatrics Society*, pages 721–727.

[Klement, 2014] Klement, N. (2014). *Planification et affectation de ressources dans les réseaux de soins : Analogie avec le probleme du bon packing. Proposition de méthodes approchées*. PhD thesis, Université Blaise Pascal.

[Konrad et al., 2010] Konrad, T., Link, C., Shackelton, R., Marceau, L., von dem Knesebeck, O., Siegrist, J., Arber, S., Adams, A., and McKinlay, J. (2010). It's about time : physicians' perceptions of time constraints in primary care medical practice in three national healthcare systems. *Med Care*.

[Kurniati et al., 2016] Kurniati, A. P., Johnson, O., and Hogg, D. (2016). Process mining in oncology : A literature review. *International Conference on Information Communication and Management*, pages 291–297.

[Lang et al., 2008] Lang, M., Burkle, T., Laumann, S., and Prokosch, H.-U. (2008). Process mining for clinical workflows : challenges and current limitations. *Studies in health technology and informatics*, pages 229–236.

[Leemans and van der Aalst, 2014] Leemans, M. and van der Aalst, W. M. P. (2014). Discovery of frequent episodes in event logs. In *International Symposium on Data-Driven Process Discovery and Analysis*, pages 1–31. Springer.

[Leemans and van der Aalst, 2015] Leemans, M. and van der Aalst, W. M. P. (2015). Process mining in software systems : discovering real-life business transactions and process models from distributed systems. In *Proceedings of the ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems*, pages 44–53. IEEE.

[Leemans et al., 2013] Leemans, S. J., Fahland, D., and van der Aalst, W. M. P. (2013). Discovering block-structured process models from event logs - a constructive approach. *International Conference on Applications and Theory of Petri Nets and Concurrency*, pages 311–329.

[Leotta et al., 2015] Leotta, F., Mecella, M., and Mendling, J. (2015). Applying process mining to smart spaces : Perspectives and research challenges. In *International Conference on Advanced Information Systems Engineering*, pages 298–304. Springer.

[Lewis, 2011] Lewis, T. G. (2011). *Network science : Theory and applications*. John Wiley & Sons.

[Liu and Qu, 2012] Liu, M. and Qu, J. (2012). Mining high utility itemsets without candidate generation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 55–64.

[Lui et al., 2006] Lui, Y., Liao, W.-K., and Choudhary, A. (2006). A two-phase algorithm for fast discovery of high utility itemsets. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 689–695.

[Maggi et al., 2013] Maggi, F. M., Dumas, M., García-Bañuelos, L., and Montali, M. (2013). Discovering data-aware declarative process models from event logs. In *Business Process Management*, pages 81–96. Springer.

[Maggi et al., 2011] Maggi, F. M., Mooij, A. J., and van der Aalst, W. M. P. (2011). User-guided discovery of declarative process models. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining*, pages 192–199. IEEE.

[Mannhardt et al., 2015] Mannhardt, F., de Leoni, M., and Reijers, H. A. (2015). The multi-perspective process explorer. In *International Conference on Business Process Management Demonstrations Sessions)*, volume 1418, pages 130–134. CEUR-WS.org.

[Mannhardt et al., 2016] Mannhardt, F., de Leoni, M., Reijers, H. A., and van der Aalst, W. M. P. (2016). Decision mining revisited-discovering overlapping rules. In *International Conference on Advanced Information Systems Engineering*, pages 377–392. Springer.

[Mannhardt et al., 2017] Mannhardt, F., de Leoni, M., Reijers, H. A., and van der Aalst, W. M. P. (2017). Data-driven process discovery-revealing conditional infrequent behavior from event logs. In *International Conference on Advanced Information Systems Engineering*, pages 545–560. Springer.

[Mannila et al., 1997] Mannila, H., Toivonen, H., and Verkamo, A. I. (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, pages 259–289.

[Mans et al., 2012] Mans, R., Reijers, H., van Genuchten, M., and Wismeijer, D. (2012). Mining processes in dentistry. *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, pages 379–388.

[Object Management Group, 2011] Object Management Group (2011). Notation (BPMN) version 2.0. *OMG Specification*.

[Papon, 2016] Papon, P.-A. (2016). *Extraction optimisée de règles d'association positives et négatives intéressantes*. PhD thesis, Université Blaise Pascal.

[Pei et al., 2004] Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. (2004). Mining sequential patterns by pattern-growth : The prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, pages 1424–1440.

[Pesic et al., 2007] Pesic, M., Schonenberg, H., and Van der Aalst, W. M. (2007). Declare : Full support for loosely-structured processes. In *11th IEEE International Enterprise Distributed Object Computing Conference*, pages 287–287. IEEE.

[Pesic and Van der Aalst, 2006] Pesic, M. and Van der Aalst, W. M. P. (2006). A declarative approach for flexible business processes management. In *International Conference on Business Process Management*, pages 169–180. Springer.

[Petri, 1962] Petri, C. A. (1962). *Kommunikation mit Automaten*. PhD thesis, Universität Hamburg.

[Poulymenopoulou et al., 2003] Poulymenopoulou, M., Malamateniou, F., and Vassilacopoulos, G. (2003). Specifying workflow process requirements for an emergency medical service. *Journal of Medical Systems*, pages 325–335.

[Prodel, 2017] Prodel, M. (2017). *Modélisation automatique et simulation de parcours de soins à partir de bases de données de santé*. PhD thesis, École Nationale Supérieure des Mines de Saint-Étienne.

[Rais and Viana, 2011] Rais, A. and Viana, A. (2011). Operations research in healthcare : a survey. *International Transactions in Operational Research*, pages 1–31.

[Rebuge and Ferreira, 2012] Rebuge, A. and Ferreira, D. R. (2012). Business process analysis in healthcare environments : A methodology based on process mining. *Information Systems*, pages 99–116.

[Rochette and Rodier, 2016] Rochette, C. and Rodier, S. (2016). Parcours de santé versus soin de la personne âgée en auvergne : proposition d'une définition organisationnelle. *Management hospitalier et territoires : les nouveaux défis*, pages 101–120.

[Rodier, 2010] Rodier, S. (2010). *Une tentative d'unification et de résolution des problèmes de modélisation et d'optimisation dans les systèmes hospitaliers : Application au nouvel hôpital Estaing*. PhD thesis, Université Blaise Pascal.

[Rojas et al., 2016] Rojas, E., Munoz-Gama, J., Sepùlveda, M., and Capurro, D. (2016). Process mining in healthcare : A literature review. *Journal of Biomedical Informatics*, pages 224–236.

[Rovani et al., 2015] Rovani, M., Maggi, F. M., de Leoni, M., and van der Aalst, W. M. (2015). Declarative process mining in healthcare. *Expert Systems with Applications*, pages 9236–9251.

[Royer, 2014] Royer, J. (2014). *Proposition d'une méthodologie de modélisation et de réorganisation du circuit du médicament dans les pharmacies hospitalières*. PhD thesis, Université Blaise Pascal.

[Rozinat and van der Aalst, 2008] Rozinat, A. and van der Aalst, W. M. P. (2008). Conformance checking of processes based on monitoring real behavior. *Information Systems*, pages 64–95.

[Scheer, 1994] Scheer, A.-W. (1994). *Business Process Engineering : Reference Models for Industrial Enterprises*. Springer-Verlag Telos.

[Schönig et al., 2016a] Schönig, S., Cabanillas, C., Jablonski, S., and Mendling, J. (2016a). A framework for efficiently mining the organisational perspective of business processes. *Decision Support Systems*, pages 87–97.

[Schönig et al., 2016b] Schönig, S., Di Ciccio, C., Maggi, F. M., and Mendling, J. (2016b). Discovery of multi-perspective declarative process models. In *International Conference on Service-Oriented Computing*, pages 87–103. Springer.

[Shie et al., 2011] Shie, B.-E., Hsiao, H.-F., Tseng, V. S., and S., Y. P. (2011). Mining high utility mobile sequential patterns in commerce environments. In *International Conference on Database Systems for Advanced Applications*, pages 224–238.

[Song et al., 2009] Song, M., Günther, C. W., and van der Aalst, W. M. (2009). Trace clustering in process mining. *Business Process Management Workshop*, pages 109–120.

[Song and van der Aalst, 2008] Song, M. and van der Aalst, W. M. P. (2008). Towards comprehensive support for organizational mining. *Decision Support Systems*, pages 300–317.

[Soulet and Rioult, 2014] Soulet, A. and Rioult, F. (2014). Efficiently depth-first minimal pattern mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 28–39.

[Srikant and Agrawal, 1996] Srikant, R. and Agrawal, R. (1996). Mining sequential patterns : Generalization and performance improvement. In *Proceedings of the 5th International Conference on Extending Database Technology : Advances in Database Technology*, pages 3–17.

[Sztyler et al., 2015] Sztyler, T., Völker, J., Carmona Vargas, J., Meier, O., and Stuckenschmidt, H. (2015). Discovery of personal processes from labeled sensor data : An application of process mining to personalized health care. In *Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data*, pages 31–46. CEUR-WS.org.

[Tapia et al., 2004] Tapia, E. M., Intille, S. S., and Larson, K. (2004). Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive Computing*, volume 4, pages 158–175. Springer.

[Tax et al., 2016a] Tax, N., Alasgarov, E., Sidorova, N., and Haakma, R. (2016a). On generation of time-based label refinements. In *Proceedings of the 25th International Workshop on Concurrency, Specification and Programming*. CEUR-WS.org.

[Tax et al., 2015] Tax, N., Bockting, S., and Hiemstra, D. (2015). A cross-benchmark comparison of 87 learning to rank methods. *Information Processing & Management*, pages 757–772.

[Tax et al., 2016b] Tax, N., Sidorova, N., Haakma, R., and van der Aalst, W. M. (2016b). Mining local process models. In *Journal of Innovation and Digital Systems*, pages 183–196.

[Tax et al., 2016c] Tax, N., Sidorova, N., Haakma, R., and van der Aalst, W. M. P. (2016c). Event abstraction for process mining using supervised learning techniques. In *Proceedings of the SAI Intelligent Systems Conference*. Springer.

[Tax et al., 2016d] Tax, N., Sidorova, N., van der Aalst, W. M. P., and Haakma, R. (2016d). Heuristic approaches for generating local process models through log projections. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining*, pages 1–8.

[Tseng et al., 2013] Tseng, V. S., Shie, B.-E., and Wu, C.-W. (2013). Efficient algorithms for mining high utility itemsets from transactional databases. In *IEEE Transactions on Knowledge and Data Engineering*, pages 1772–1786.

[Van den Bergh et al., 2013] Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., and De Boeck, L. (2013). Personnel scheduling : A literature review. *European Journal of Operational Research*, pages 367–385.

[van der Aalst, 2016] van der Aalst, W. (2016). *Process Mining : Data Science in Action*. Springer Berlin Heidelberg.

[van der Aalst et al., 2011a] van der Aalst, W., Adriansyah, A., and Van Dongen, B. (2011a). Causal nets : A modeling language tailored towards process discovery. In *Proceedings of the 22Nd International Conference on Concurrency Theory*, pages 28–42.

[van der Aalst et al., 2011b] van der Aalst, W. M. P., Adriansyah, A., de Medeiros, A. K. A., Arcieri, F., Baier, T., Blickle, T., Bose, R. P. J. C., van den Brand, P., Brandtjen,

R., Buijs, J. C. A. M., Burattin, A., Carmona, J., Castellanos, M., Claes, J., Cook, J., Costantini, N., Curbera, F., Damiani, E., de Leoni, M., Delias, P., van Dongen, B. F., Dumas, M., Dustdar, S., Fahland, D., Ferreira, D. R., Gaaloul, W., van Geffen, F., Goel, S., Günther, C. W., Guzzo, A., Harmon, P., ter Hofstede, A. H. M., Hoogland, J., Ingvaldsen, J. E., Kato, K., Kuhn, R., Kumar, A., Rosa, M. L., Maggi, F. M., Malerba, D., Mans, R. S., Manuel, A., McCreesh, M., Mello, P., Mendling, J., Montali, M., Nezhad, H. R. M., zur Muehlen, M., Munoz-Gama, J., Pontieri, L., Ribeiro, J., Rozinat, A., Pérez, H. S., Pérez, R. S., Sepúlveda, M., Sinur, J., Soffer, P., Song, M., Sperduti, A., Stilo, G., Stoel, C., Swenson, K. D., Talamo, M., Tan, W., Turner, C., Vanthienen, J., Varvaressos, G., Verbeek, H. M. W., Verdonk, M., Vigo, R., Wang, J., Weber, B., Weidlich, M., Weijters, T., Wen, L., Westergaard, M., and Wynn, M. T. (2011b). Process mining manifesto. In *Business Process Management Workshops (1)*, pages 169–194.

[van der Aalst et al., 2012] van der Aalst, W. M. P., Adriansyah, A., and van Dongen, B. F. (2012). Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, pages 182–192.

[van der Aalst et al., 2009a] van der Aalst, W. M. P., Pesic, M., and Schonenberg, H. (2009a). Declarative workflows : Balancing between flexibility and support. *Computer Science - Research and Development*, pages 99–113.

[van der Aalst et al., 2009b] van der Aalst, W. M. P., van Dongen, B. F., Günther, C. W., Rozinat, A., Verbeek, E., and Weijters, T. A. (2009b). Prom : the process mining toolkit. *BPM Demos*, pages 1–4.

[van der Werf et al., 2008] van der Werf, J. M. E. M., van Dongen, B. F., Hurkens, C. A. J., and Serebrenik, A. (2008). Process discovery using Integer Linear Programming. In *International Conference on Applications and Theory of Petri Nets*, pages 368–387. Springer.

[Weijters et al., 2006] Weijters, T. A., van der Aalst, W. M., and de Medeiros Ana K., A. (2006). Process mining with the heuristics miner-algorithm. *Tech. Rep. WP*, pages 1–34.

[Weijters et al., 2004] Weijters, T. A., van der Aalst, W. M., and Maruster, L. (2004). Workflow mining : discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, pages 1128–1142.

[Wen et al., 2007] Wen, L., Aalst, W. M., Wang, J., and Sun, J. (2007). Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery*, pages 145–180.

[Xing and Ghorbani, 2004] Xing, W. and Ghorbani, A. (2004). Weighted pagerank algorithm. In *Proceedings of the Second Annual Conference on Communication Networks and Services Research*, pages 305–314. IEEE.

[Yan et al., 2003] Yan, X., Han, J., and Afshar, R. (2003). Clospan : Mining : Closed sequential patterns in large datasets. In *SIAM International Conference on Data Mining*, pages 166–177.

[Yang and Su, 2014] Yang, W. and Su, Q. (2014). Process mining for clinical pathway : Literature review and future directions. *11th International Conference on Service Systems and Service Management*, pages 1–5.

[Yao and Hamilton, 2006] Yao, H. and Hamilton, H. J. (2006). Mining itemset utilities from transaction databases. *Data & Knowledge Engineering*, pages 602–626.

[Yao et al., 2004] Yao, H., Hamilton, H. J., and Butz, C. J. (2004). A foundational approach to mining itemset utilities from databases. *SIAM International Conference on Data Mining*, pages 482–486.

[Yin et al., 2012] Yin, J., Zheng, Z., and Cao, L. (2012). Uspan : An efficient algorithm for mining high utility sequential patterns. In *Proceedings of the 18th International Conference on Knowledge Discovery and Data Mining*, pages 660–668.

[Yun, 2008] Yun, U. (2008). A new framework for detecting weighted sequential patterns in large sequence databases. *Knowledge-Based Systems*, pages 110–122.

[Zaki, 2000] Zaki, M. J. (2000). Scalable algorithm for association mining. In *IEEE Transactions on Knowledge and Data Engineering*, pages 372–390.

[Zaki, 2001] Zaki, M. J. (2001). Spade : An efficient algorithm for mining frequent sequences. *Machine Learning*, pages 31–60.

[Zeising et al., 2014] Zeising, M., Schönig, S., and Jablonski, S. (2014). Towards a common platform for the support of routine and agile business processes. In *Collaborative Computing : Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference on*, pages 94–103. IEEE.

[Zida et al., 2017] Zida, S., Fournier-Viger, P., Lin, J. C.-W., Wu, C.-W., and Tseng, V. S. (2017). Efim : A fast and memory efficient algorithm for high-utility itemset mining. *Knowledge and Information Systems*, pages 595–625.

[Zida et al., 2015] Zida, S., Fournier-Viger, P., Wu, C.-W., Lin, J. C., and Tseng, V. S. (2015). Efficient mining of high utility sequential rules. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 157–171.

[zur Muehlen and Recker, 2008] zur Muehlen, M. and Recker, J. (2008). How much language is enough ? theoretical and practical use of the business process modeling notation. *International Conference on Advanced Information Systems Engineering*, pages 465–479.