No d'ordre 2007-ISAL-0039 Année 2007

Thèse

Contributions aux techniques de Prise de Décision et de Valorisation Financière

Présentée devant
L'Institut National des Sciences Appliquées de Lyon

Pour obtenir
Le grade de docteur

École doctorale : Informatique et Information pour la Societé (EDIIS)

par

Hunor ALBERT-LORINCZ
(ingénieur)

soutenue publiquement le 6 juillet 2007 devant la commission d'examen

Jury MM.

Guillaume BESLON		Maître de Conférences à l'INSA Lyon
Jean-François BOULICAUT	Directeur	Professeur à l'INSA Lyon
Arnaud GIACOMETTI	Rapporteur	Professeur à l'Université de Tours
Bart GOETHALS		Chercheur à l'Université d'Anvers (B)
Alain MILLE		Professeur à l'Université Lyon 1
Pascal PONCELET	Rapporteur	Professeur à l'Ecole des Mines d'Ales
Denis WALLEZ		Nomura International plc., Londres (UK)

Cette thèse a été préparée au Laboratoire d'InfoRmatique en Image et Systèmes d'information (UMR CNRS 5205) à l'INSA de Lyon

Remerciements

Un grand merci à mon directeur thèse, Jean-François Boulicaut, Professeur à l'INSA de Lyon, pour son soutien moral et scientifique continue depuis le début de mon DEA en 2002.

Je suis extrêmement reconnaissant à mes deux rapporteurs, Professeurs Pascal Poncelet et Arnaud Giacometti. Leurs observations et conseils ont grandement contribué à améliorer la qualité et la présentation des résultats de la thèse.

Je remercie le président du jury, le Professeur Alain Mille pour l'intérêt qu'il a porté à mon travail, et Bart Goethals pour toutes les discussions et idées de développement exprimées pendant la soutenance.

J'exprime un remerciement spécial à Denis Wallez, ex-responsable du groupe *Modèles et Méthodologies* de Nomura International plc., qui m'a permis de travailler sur des sujets de recherche passionnants dans son équipe.

Je désire remercier mon ancienne équipe *Structurés de Taux* au Crédit Mutuel-CIC. Merci à Xavier Baudoin, Tarek Belkahia et Laurent Altmeyer de m'avoir formé aux marchés financiers et veillé sur mes premiers pas dans ce nouveau monde.

Je suis heureux d'avoir rencontrer au Liris des chercheurs et doctorants talentueux : Christophe Rigotti, Claire Leschi, Jérémy Besson, Céline Robardet, Ieva Mitasiunaite, Cyrille Masson, Ruggero G. Pensa, Loic Cerf et Nicolas Méger.

Table des matières

Table des figures

Index

Préambule

Ce doctorat a été préparé dans le cadre d'un travail académique au sein du laboratoire LIRIS (équipe "Data Mining", 2002-2003) mais aussi dans deux institutions financières différentes, la salle des marchés du groupe CIC (Paris, 2004-2005) et un groupe de modélisation de la banque Nomura International plc. (Londres, depuis 2006). Mes objectifs initiaux consistaient à travailler sur les applications de nouveaux algorithmes d'extraction de motifs séquentiels dans le domaine de la finance. Il s'agissait, par exemple, de traiter des cas difficiles d'extraction de motifs sous contraintes (e.g., le calcul de toutes les séquences fréquentes et satisfaisant une expression régulière donnée dans une grande collection de séquences). Nous pensions alors qu'il serait possible d'exploiter de tels algorithmes dans des contextes financiers originaux. Cependant, ces objectifs ont laissé la place à des développements plus en accord avec les besoins pointus des entreprises qui m'ont employé (amélioration des indicateurs techniques pour construire des stratégies de "trading"[1], amélioration de l'efficacité des techniques de simulations Monte Carlo doubles utilisées pour la prise de décision sur des produits financiers avec options de rappel).

Mon mandat dans le groupe Models et Méthodologies de Nomura est très ouvert : assumer la veille technologique et la recherche "in house" nécessaires pour assurer le standard de qualité des investissements de la banque. Ainsi, je me suis intéressé à l'amélioration des modèles de valorisation des produits dérivés, en proposant des simulations plus précises ou plus rapides que les techniques existant chez Nomura. Le calcul de motifs séquentiels n'a pas été utilisé dans ces développements.

Je ne souhaite donc pas argumenter d'un fil conducteur, nécessairement artificiel, qui permettrait d'expliquer comment mes différents travaux servent un objectif unique. Ceci étant, mes contributions sont toutes liées à l'exploitation de données pour aider à la prise de décision, qu'il s'agisse du traitement de données expérimentales ou historisées ou bien de données issues de processus de "modélisation avec simulation". Nous proposons une contribution significative dans le domaine des algorithmes d'extraction de motifs dans des collections de séquences, c'est-à-dire un travail assez typique pour un chercheur en "data mining". Cette proposition peut s'appliquer à de nombreuses tâches d'analyse qui peuvent exploiter la connaissance de régularités dans des séquences, par exemple, des séries temporelles comme des historiques de cours. Nous décrivons ensuite nos deux applications financières qui, tout en exploitant des méthodes de fouille de données ou d'apprentissage statistique, ne font pas usage de nos développements algorithmiques originaux. Ce faisant, nous cherchons tout de même à dégager les aspects génériques de chacune de ces applications pour mettre en valeur l'impact possible de notre travail.

[1] Les derniers chapitres de la thèse font mention aux termes et concepts financiers. Le milieu de la finance les utilise en anglais, la traduction française étant inexistante ou imprécise - et ne serait pas reconnue par le lecteur financier. Nous ferons cependant l'effort d'employer des guillemets pour les premières apparitions de ces anglicismes, et proposons un glossaire.

Chapitre 1

Introduction

1.1 Le contexte

Il est chaque jour plus facile de collecter des données, mais notre capacité à en extraire des informations à forte valeur ajoutée reste limitée. Dans la plupart des secteurs d'activités, que ce soit dans le domaine scientifique, le commerce ou les secteurs bancaires et industriels, d'énormes volumes de données sont disponibles et à ce jour encore très peu exploités. Dans le domaine scientifique, nous pourrions prendre des exemples dans les données accumulées par les agences spatiales. Le commerce, a fortiori lorsqu'il est électronique, fournit des données précieuses sur les clients et leurs comportements. Les banques doivent, notamment, analyser l'état du marché pour prendre rapidement de bonnes décisions quant aux investissements et aux caractéristiques des produits financiers qu'elles proposent. Notons d'ailleurs que dans ce contexte applicatif qui nous intéresse, les transactions boursières sont archivées et gardées sur des périodes de plusieurs années, notamment du fait de réglementations qui obligent les acteurs économiques à garder des traces de leurs activités. Enfin, des industries comme, par exemple, celle des télécommunications, produisent des masses de données qui contiennent, implicitement, des informations importantes pour améliorer la qualité des services. En d'autres termes, les défis ne sont plus de collecter des données mais bien d'en faire de la valeur ajoutée.

Pour répondre à ces opportunités, l'extraction de connaissances dans les bases de données (ECBD ou "Knowledge Discovery in Databases", parfois appelé "Data Mining") est le domaine de recherche au sein duquel coopèrent statisticiens, spécialistes en bases de données et en intelligence artificielle (notamment en apprentissage automatique), ou encore chercheurs en conception d'interfaces homme-machine. Ce domaine connaît une croissance spectaculaire, sous l'impulsion des organisations propriétaires de données. Le domaine de recherche de l'ECBD est vaste et encore mal délimité. Cependant, quelques bon ouvrages de synthèse sont disponibles. Le fameux ouvrage de Fayyad et al. [40] commence à dater puisque la plupart des contributions qu'il contient sont représentatives de l'état de l'art au milieu des années 90. Parmi les publications plus récentes, nous recommandons [48, 52].

L'extraction de connaissances dans les bases de données a été défini dans [42] comme *le processus non trivial d'extraction d'informations valides, nouvelles, potentiellement utiles, et compréhensibles à partir de données.* Les informations extraites, qui conduisent parfois à l'explicitation de connaissances du point de vue des propriétaires des données, peuvent

être de natures variées (motifs exprimant des régularités, règles descriptives ou règles de classification, arbres ou graphes, ensemble d'objets similaires, etc). Il existe de nombreuses discussions sur les différentes phases de tels processus (voir notamment [40]). Nous n'insistons ici que sur quelques caractéristiques fondamentales de ce type de processus.

- *Une succession d'étapes*
  Ces processus complexes comportent plusieurs étapes que l'on peut résumer en 5 macro-étapes : la *sélection* des sources de données brutes, la *préparation* de ces données, *l'extraction* proprement dite, le *post-traitement* des propriétés découvertes et enfin *l'exploitation* des résultats. Préparer les données, ce peut être construire une collection de séquence d'événements discrets à partir de données brutes fournies par, par exemple, l'évolution de cours boursiers. Extraire, ce peut être de calculer toutes les séquences d'événements qui sont suffisamment fréquentes et qui précède toujours un événement particulier (i.e., une recherche de signature). Post-traiter, cela peut être d'éliminer des successions d'événements dont l'explication de la présence est soit triviale soit bien connue des experts. Exploiter cela peut être de mettre en place des alarmes spécifiques lorsque les signatures de l'événement particulier sont repérées. Exploiter ce sera souvent prendre en compte les résultats pour itérer sur de nouvelles extractions.
- *Un processus itératif*
  Chaque étape peut conduire à la remise en cause de choix effectués lors des étapes précédentes. Par exemple, la découverte de propriétés majoritairement triviales lors d'une extraction aura un impact sur l'étape de sélection des données pour de nouvelles extractions. De plus, il est rare qu'un processus d'extraction ne puisse reposer que sur une seule technique de fouille (et donc un type de propriété).
- *Un processus interactif*
  Les processus ECBD sont intrinsèquement non-automatiques. L'utilisateur doit intervenir lors de chaque étape pour exploiter sa connaissance du domaine et des besoins d'analyse dans la sélection et la préparation des données, le choix de la bonne technique d'extraction, le paramétrage des extracteurs, la validation de la pertinence des propriétés trouvées, l'utilisation de celles-ci ou encore la décision d'itérer. Il faut veiller à aller au bout des processus de découverte de connaissances. Ainsi, pour conclure (et rémunérer une prestation de découverte des connaissances), il conviendra de faire le bilan de l'impact des décisions basées sur ces nouvelles connaissances. Dans les applications économiques qui nous intéressent, cette mesure d'impact pourra être chiffrée.

Notre travail se situe dans le contexte général de la *prise de décision* financière. Le terme est à prendre dans un sens très étendu qui comprends : la sélection de candidats optimisant une fonction cible (e.g. "stock picking"), l'apprentissage des attributs qui caractérisent une situation favorable (e.g., un apprentissage automatique supervisé) ou la classification non supervisée pour regrouper des situations/cas similaires ("clustering") et, a contrario, faciliter l'identification d'anomalies. De manière générale, nous étudions la maximisation d'une fonction d'utilité dont la valeur dépendra de la réalisation de l'un des états possibles du monde : il s'agit de prévoir la valorisation de certains choix. Dans ce cadre probabiliste, le décisionnaire ne connait pas la future réalisation du monde, mais il cherche à mieux caractériser le présent afin de construire de meilleures prévisions. Ces connaissances peuvent le conduire à limiter le champs des possibilités, à exclure des is-

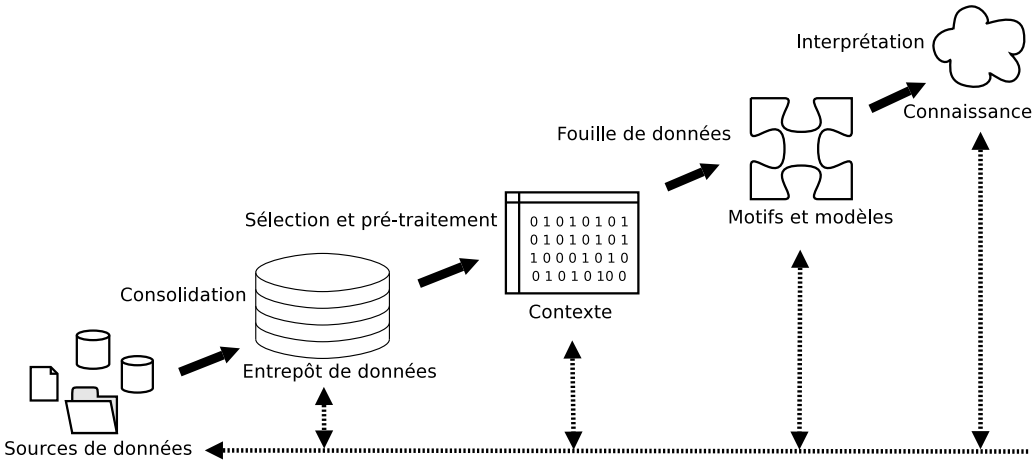


FIG. 1.1 – *Le processus d'extraction de connaissances à partir des données.*

sues improbables et donc à avoir de meilleures stratégies économiques, autrement dit une amélioration du rendement.

En finance, les états du monde (événements) sont souvent agrégés dans des *séries temporelles* qui contiennent les marques de très nombreuses dépendances $Cause \rightarrow Effet$. Suivant l'intuition que l'amélioration des décision serait possible en étudiant les causalités dans les séries temporelles, nous nous sommes intéressés à l'extraction et à la manipulation des données séquentielles. D'autres techniques seront également utilisées dans les applications concrètes, mais notre contribution théorique et algorithmique concerne principalement la découverte de motifs dans des données séquentielles.

Ces contextes financiers induisent des problématiques de collecte et de pré-traitement des données spécifiques, des interactions avec des acteurs diversifiés, et notamment les experts dans le domaine économique visé par la prise de décision (e.g., un "trader"). Par exemple, au delà des classiques problèmes de traitement des valeurs manquantes ou encore extrêmes, les historiques des cours boursiers doivent être retraités en tenant compte des augmentations de capitaux, et de rachats ou scissions d'actions : oublier que chaque action a donné naissance à trois autres, ou que 6% du flottant a été racheté par une compagnie peut conduire à des analyses éronnées et faire échouer la découvertes des connaissances. Notons également que le "reporting" des compagnies contient souvent un décalage entre les causes et les effets [1]. Pour exploiter les données par des méthodes discrètes, comme par exemple, celles que nous avons développé, il faut transformer (e.g., discrétiser, quantiliser) les données continues.

1.2 Contribution à la fouille de données et la finance

En tant qu'activité à forte valeur ajoutée, la finance a été très (et certainement trop) enthousiaste pour les possibilités offertes par l'apprentissage automatique. L'engouement

[1] Les bourses européennes réagissent, en général, aux annonces de la veille des bourses américaines, et deux données du même jour de ces deux bourses peuvent donc être liés par une relation de cause à effet. Supposer la concomitance serait ici une grave erreur.

des années 90 pour les arbres de décision (avec, par exemple, la méthode CART [26]) et les réseaux de neurones (utilisés pour la construction de classifieurs) n'a pas duré bien longtemps. Confronté à des problèmes trop difficiles pour l'époque, l'engouement initial a cédé la place au scepticisme. Ceci est également due à l'école de Fama, rejointe plus tard par Tobin, avec sa théorie des *marchés efficients* [37] selon laquelle "toute l'information historique est déjà intégrée dans les prix des actifs". Dans la finance moderne, bâtie sur les travaux de Black, Scholes et Merton [22, 74], les prix des actifs normalisés (cf. chapitre 6) sont des martingales, et toute tentative de prédiction de cours futurs devrait échouer.

Ce n'est qu'avec l'essor des "hedge funds", et l'accroissement des moyens informatiques que certains changements subtils ont commencé à être visibles. Ainsi, on assiste à la naissance de la finance comportementale ("behavioural finance") auquel Fama finit par adhérer [39]. Ainsi, certaines banques développent des automates de "trading" qui surveillent les marchés et génèrent des transactions. De tels types de travaux n'ont pas été immédiatement reconnus mais il n'est plus possible d'ignorer que les institutions qui recourent à l'apprentissage automatique afin d'améliorer leurs stratégies d'investissement ont obtenu des résultats tangibles. Aujourd'hui, l'efficience des marchés redevient donc un objet de controverse.

Ce mémoire développe deux types de contributions. Tout d'abord, nous considérons le problème difficile mais très étudié de l'extraction de motifs séquentiels devant satisfaire une conjonction de contraintes. Nous introduisons la famille de contraintes bien partitionnées qui permettent une décomposition hiérarchique de l'espace de recherche et son parcours très efficace ("cascaded constraint miner"). Le cas particulier de la conjonction d'une contrainte de fréquence minimale et d'une contrainte d'expression régulière (éventuellement généralisée, i.e., une phrase d'un langage sensible au contexte ou contenant des variables type aXbX) est décrit en détail. Ces travaux ont été décrits initialement dans [5, 6] pour le cadre RE-Hackle et dans [7] pour sa généralisation. Cette contribution algorithmique ouvre la voie vers des stratégies d'élagages adaptatives et permettent ainsi de trouver des compromis entre l'exploitation active de contraintes non anti-monotones (typiquement les expressions régulières) sans pour autant pénaliser les remarquables possibilités d'élagage des contraintes anti-monotones (comme typiquement la fréquence minimale). Nous comparons donc notre proposition vis-à-vis du travail pionnier sur les algorithmes Spirit [44] qui, tout en ayant identifié le problème, proposait aux experts de choisir des relaxations des contraintes d'expressions régulières pour atteindre des compromis ad-hoc. Avec RE-Hackle, nous ne demandons pas d'arbitrage humain entre les élagages sur la fréquence ou ceux sur les contraintes syntaxiques : la stratégie peut évoluer dynamiquement pendant l'extraction.

Sans pour autant pouvoir développer des applications des algorithmes développés pendant notre première partie de préparation de thèse, nous avons ensuite travailler à des applications originales dans le secteur de la finance. Dans une première application, nous proposons d'améliorer le pouvoir prédictif des indicateurs techniques en introduisant des signatures pour les configurations de marché et ainsi améliorer les performances des automates de "trading". Les signatures sont ici déterminées à l'aide du calcul d'ensembles fréquents, i.e., une des techniques phares en fouille de données. Les gains de performances sont statistiquement significatifs et ce travail a été présenté dans [8].

Nous avons également travaillé à la valorisation d'une classe particulière de produits dérivés (les options bermudéennes) où l'une des parties a le droit de terminer le contrat à

une série de dates prédéterminées. Au cours des développements, il s'est avéré nécessaire de calculer l'espérance conditionnelle des futurs *cash flows* dans un futur état de monde. Traditionnellement, cela était possible par exemple en utilisant une nouvelle simulation Monte Carlo, i.e., une technique très gourmande en temps de calcul. Afin de traiter ce sous-problème d'une manière efficace, nous avons développé une nouvelle méthode baptisée neighbourhood Monte Carlo que l'on a trouvé jusqu'à 20 fois plus rapide que les méthodes précédentes. On peut alors linéariser les Monte Carlo doubles dans les situations qui n'ont pas besoin d'une très grande précision. Les résultats sont présentés dans le rapport de recherche [9], un rapport actuellement en cours d'évaluation pour un journal en finance computationnelle.

1.3 Plan du mémoire

Ce mémoire comporte deux parties. La première partie (chapitres 2 à 4) concerne l'extraction de motifs séquentiels sous différentes formes de contraintes. Elle correspond à notre contribution algorithmique. Le chapitre 2 est un état de l'art sur les algorithmes existants pour l'extraction de motifs sous contraintes. Le chapitre 3 propose un cadre générique pour des processus de génération de motifs candidats par des parcours ascendants dans des arbres qui structurent les espaces de recherche. Le chapitre 4 décrit une instance particulière au cas des contraintes d'expressions régulières, éventuellement généralisées.

La seconde partie (chapitres 5 à 6) présente nos applications dans le domaine de la finance. Le chapitre 5 applique les motifs ensemblistes fermés fréquents pour attribuer des signatures aux configurations de marché afin d'améliorer le pouvoir prédictif des indicateurs techniques, et donc les stratégies d'investissements. Le chapitre 6 constitue une incursion dans le domaine de la valorisation d'une famille de produits financiers très populaires : les options de rappel.

Enfin, un chapitre de conclusion résume nos contributions et indique quelques unes de leurs perspectives.

Première partie

Données séquentielles et motifs sous contraintes

Introduction

Les données séquentielles sont omniprésentes, qu'il s'agisse de données ordonnées dans le temps ou dans l'espace. Savoir repérer des événements dans le temps ou des régularités spatiales sont des fonctionnalités très bien développées chez l'homme, pourvu que les données spatio-temporelles considérées ne soient pas trop nombreuses. Quand la taille des collections de données augmente, l'homme a besoin des techniques automatisées et d'un traitement instrumentalisé de l'information. La *fouille de données* ("data mining") répond à ce besoin, et fournit une collection d'outils, i.e., principalement des algorithmes, permettant de trouver des motifs et des structures présentes dans les données. Ces *motifs* sont, par exemple, des successions contiguës ou non d'éléments ou d'ensembles d'éléments.

Qu'il s'agisse du traitement des données temporelles ou spatiales, la partie algorithmique est quasi-identique. Le domaine d'application comprend ainsi l'analyse des données biologiques (successions d'amino-acides, protéines, etc.) [11, 17, 103], le "text mining" [59], l'étude des chemins de navigation WWW [30, 31, 57] ou encore l'analyse de propagation des alarmes dans les réseaux de télécommunications [62].

La recherche des motifs séquentiels fréquents dans une base de données des séquences est une tâche très étudiée (voir, par exemple, [3, 97, 71, 107, 90]). Alors que les premiers extracteurs devaient trouver les séquences fréquentes, les chercheurs ont vite reconnu l'importance de pouvoir intégrer d'autres contraintes. Dans la plupart de ces applications, l'impossibilité que l'utilisateur spécifie des contraintes pour restreindre l'information à seulement ce qui l'intéresse, conduit à des phases de post-traitement coûteux. La spécification des contraintes additionnelles à la contrainte de fréquence en amont de l'extraction a donc été très utile [44, 97, 106]. Ceci limite le nombre de séquences inutilement reportées à l'utilisateur, et peut potentiellement réduire les temps d'extraction, à condition que la conjonction des contraintes soit traitable au cours de la génération des candidats.

Le chapitre suivant dresse un *État de l'art* de l'extraction des séquences et considère les algorithmes génériques qui s'appliquent sur une grande variété de séquences et contraintes. Les adaptations/solutions ad-hoc spécifiques à un seul problème ponctuel ne seront pas incluses dans cette étude. Uniquement les *méthodes déterministes complètes*, qui trouvent toujours l'ensemble des motifs séquentiels satisfaisant les contraintes imposées seront présentées.

Un nombre de publication important étudie les situations ou la taille de la base de données augmente pendant l'extraction. Il s'agit de la découverte incrémentale [76, 81, 46, 36, 109, 101] et c'est un domaine de recherche très actif. Dans leur état de l'art [77], Masseglia et al. ont identifié deux tendances : certaines équipes cherchent à stocker des informations supplémentaires au cours de l'extraction des séquences (e.g., une frontière négative [81, 109]). Un second courant [76, 101] ne conserve que les séquence fréquentes, et revisite la base chaque fois qu'un ajout de nouvelles transactions pourrait diminuer le support des séquences existantes, ou rendre fréquent de nouvelles séquences.

Structurellement, notre *État de l'art* présente d'abord les notations et définitions uti-

lisées tout au cours de ce mémoire. Cette partie est suivie d'une discussion des propriétés des contraintes et de leurs impacts sur l'extraction. La majeure partie de cette étude concerne la présentation de quatre familles d'algorithmes : la classe Apriori, les techniques de projections, les listes d'occurrences, et les espaces des versions.

Nous nous intéressons à la spécification conjointe de plusieurs types de contraintes (conjonction de contraintes primitives). Nous étudions tout particulièrement des formes riches de contraintes syntaxiques comme, par exemple, les contraintes qui restreignent la forme possible des motifs par des expressions régulières.

Chapitre 2

Définitions et Etat de l'Art

Depuis les travaux d'Agrawal et Srikant [3], l'extraction des motifs séquentiels a reçu beaucoup d'attention. Mannila et al. introduisent Winepi et Minepi pour l'extraction des séquences à partir une série d'événements [69, 70, 71]. Elles traitent les épisodes séries et parallèles. Dans les deux cas, les données sources de l'extraction sont constituées d'une séquence d'événements. On mentionnera aussi GSP [97] qui est l'adaptation aux séquences de l'algorithme Apriori [2]. Spade [107], FreeSpan [49] ou PrefixSpan [90] sont des propositions plus récentes.

L'extraction des séquences (avec répétition des événements) est un problème très difficile. Si plusieurs techniques algorithmiques ont permis l'extraction efficace des ensembles fréquents dans des données transactionnelles [88, 25, 49, 28], ces résultats ne se transposent pas facilement à l'extraction des motifs séquentiels. L'ordre des éléments d'une séquence doit être conservé, ce qui demande des efforts supplémentaire par rapport aux ensembles fréquents. Dans le cas des ensembles, l'espace de recherche a été limité par l'ensemble vide comme borne inférieure et le motif contenant tous les symboles de l'alphabet comme borne supérieure. Dans le cas des séquences, un symbole peut apparaître moultes fois au sein de la même séquence : l'espace de recherche est infini et l'extracteur doit préserver l'information de précédence des symboles. Les techniques efficaces de comptage basées sur le tri des items (ordre lexicographique initiale ou alternée par la fréquence des items [20] ou le comptage horizontal ne sont pas adaptées aux séquences. Il est très difficile d'extraire des motifs longs [20], et le nombre de candidats croit exponentiellement en fonction de ta taille des ensembles (ou longueur des séquences). Plusieurs auteurs proposent des structures de données efficaces, par exemple [33], mais la complexité intrinsèque reste non polynomiale. [96] est une synthèse sur la recherche de séquences. De très nombreux chercheurs ont permis d'améliorer considérablement l'efficacité des extracteurs de séquences [35, 64, 90, 91, 101, 105, 106, 107]. Cependant, le manque de pertinence des séquences retournées est devenu un problème qui, à son tour, a catalysé beaucoup d'efforts. En effet, la sortie d'un extracteur est peu intelligible pour un utilisateur humain qui se retoruve noyé dans une quantité d'informations dépassant souvent de façon spectaculaire la taille de la base de séquences de départ. L'identification des motifs intéressant l'expert est un problème important. La restriction de la sortie à des sous-ensembles a priori plus intéressants peut prendre la forme d'un filtrage a postériori (post-traitement de la sortie de l'extracteur). Alternativement, elle peut souvent être spécifiée de manière interactive pendant l'extraction : l'intérêt subjectif de l'expert peut alors s'exprimer sous la forme

de contraintes que l'extracteur doit pouvoir essayer d'exploiter pendant les calculs. La première approche (filtrage) est donc peu efficace et la seconde fait l'objet de nombreux développements. La section 2.2 revoit les principaux types de contraintes que la recherche a introduit, ainsi que leur propriétés qui rendent l'extraction plus facile ou plus difficile.

2.1 Définitions

La **chaine de longueur nulle** et **l'ensemble vide** seront notés ϵ et respectivement ϕ. La **séquence** $S = s_1 s_2 \dots s_i \dots s_n$ est une suite d'évènements s_i entre lesquels il existe une relation de précédence temporelle (ou spatiale). Le i^{ieme} **élément** de S correspond à l'événement à la i^{ieme} position spatio-temporelle. Si les éléments de S sont pris dans un **alphabet** de symboles (items) Σ, i.e., $s_i \in \Sigma = \{a_1 a_2 \dots a_m\}$ on parle de **séquences de symboles**. Σ^* dénote l'ensemble des séquences possibles sur l'alphabet Σ. Un **épisode** est un ensemble d'évènements $\{e_1, e_2, \dots e_n\}, e_1, e_2, \dots e_n \in \Sigma$ entre lesquels il n'y a pas de relation de précédence spatio-temporelle. Nous disons que $e_1, e_2, \dots e_n$ sont concomitants. Si les éléments d'une séquence sont des épisodes, on parle de **séquence d'épisodes**, i.e., $s_i = \{s_1^i, s_2^i, \dots, s_{m(i)}^i\}$, avec $\forall j \in \{1 \dots m(i)\}, s_j^i \in \Sigma$. Dans la suite, le terme séquence désignera une séquence de symboles.

Soit $S \in \Sigma^*$, une séquence quelconque. La **longueur** de S, noté $\|S\|$, correspond au nombre d'éléments dans une séquence. Nous utilisons des majuscules pour les motifs séquentiels et des minuscules pour leurs éléments (symboles de Σ). Une séquence de longueur k est appelé une k-séquence.

Une séquence X est **sous-séquence** d'une séquence Y s'il existe un indice p à partir duquel $x_i = y_{p+i}$ et $i <= \|X\|$, et $\|X\| \leq \|Y\|$, (une séquence n'est pas sous sous-séquence d'elle même).

La relation binaire *sousSéquence(X,Y)* est définie comme : $X \subseteq Y$ ssi $\exists p$, tel que $\forall i <= \|X\| \Rightarrow x_i = y_{p+i}$ (Il faut aussi $\|X\| < lenY + p$). On distingue deux cas spécifiques. Si $p = 0$, X est **préfixe** de Y (Noté $X \prec Y$). Si $p = \|Y\| - \|X\|$, la sous-séquence X est appelée **suffixe** de Y.

Une séquence de longueur k est appelée k-séquence, alors qu'un préfixe ou suffixe de longueur k est un k-préfixe ou k-suffixe. Par exemple ab est un 2-préfixe de $abdef$.

Une séquence de symboles $X = x_1 x_2 \dots x_n$ est appelée **sous-séquence creuse** de $Y = y_1 y_2 \dots y_m$ s'il existe une suite d'entiers croissants $1 \leq i_1 \leq i_2 \dots i_m \leq n$ tel que $x_1 = y_{i1}, x_2 = y_{i2} \dots x_m = y_{im}$. Par exemple, *ace* est sous-séquence creuse de *cabace*. On notera $X \triangleleft Y$.

La **base de données des séquences** $D = \{D_1, D_2 \dots D_n\}$ est un ensemble de séquences où chaque D_k est une séquence bâtie sur l'alphabet Σ.

Les contraintes considérées sont des fonctions $C : (S \in \Sigma^*, D) \rightarrow \{vrai, faux\}$. Si $C(S, D)$ est vrai (resp. faux), on dit que la séquence satisfait (resp. viole) la contrainte C.

Le **support** d'une séquence S est défini comme $support(S) = \frac{\|\{T \in D \text{ et } S \subseteq T\}\|}{\|\{T \in D\}\|}$. On peut exiger que le support d'une séquence dépasse un certain seuil ($support(S) > minsup$). On parle alors d'une contrainte de fréquence.

L'extraction sous une conjonction de contraintes $C = C_1 \wedge C_2 .. \wedge C_n$ est la recherche de toutes les séquences S tel que $\exists T \in D$ et $S \subseteq T$ et $\forall i \in 1 \dots n \Rightarrow C_i(S) = vrai$. Men-

tionnons la contrainte de fréquence, diverses contraintes temporelles [107], les contraintes exprimées par des expressions régulières [44] ou encore les contraintes de similarité [34], [82]. Une approche naïve pour gérer la présence de contraintes consisterait à générer toutes les séquences possibles et filtrer celles qui satisfont la conjonction. Il s'agit d'une approche de post-traitement (ou encore *générer et tester*), qui a un coût considérable. La tendance actuelle est de *pousser* les contraintes le plus loin possible à l'intérieur des algorithmes d'extraction afin de ne générer au plus tôt que des séquences dites intéressantes. En fonction de leurs propriétés, les contraintes sont plus ou moins faciles à pousser à l'intérieur du processus d'extraction. Par la suite, nous considérons que la contrainte de fréquence est toujours présente.

Les opérateurs suivants de manipulation de séquences seront très utiles dans les chapitres suivants.

La **concaténation k-télescopée** de deux séquences S et P ($Op \oplus_k (S, P)$ ou $S \oplus_k P$) est une séquence. Les k derniers symboles de S doivent coïncider avec les k premiers symboles de P. Si k est zéro ou négatif, on obtient la concaténation usuelle.

$$S \oplus_k P = \begin{cases} \text{si } k > 0,\ k < \|S\|,\ k < \|P\| \text{ et } (\forall i > 0, i \leq k \Rightarrow P_i = S_{\|S\|-k+i}) \\ \qquad s_1 s_2 \ldots s_{\|S\|-k} p_1 p_2 p_{\|P\|} \\ sinon \\ \qquad \epsilon \end{cases}$$

Par exemple $abc \oplus_2 bcdef = abcdef$, $abc \oplus_2 cdef = \epsilon$ et $ab \oplus_0 df = abdf$.

La concaténation k-télescopée de deux ensembles de séquences $\{S_1, S_2, \ldots S_n\}$ et $\{P_1, P_2, \ldots P_m\}$ est la concaténation de toutes les séquences dans leur produit cartésien :

$$\{S_1, S_2, \ldots, S_n\} \oplus_k \{P_1, P_2, \ldots, P_m\} = \{S_i \oplus_k P_j \mid 0 < i \leq n, 0 < j \leq m\}$$

La concaténation k-télescopée de n séquences $\{S_1, S_2, \ldots S_n\}$ est

$$Op \oplus_k (S_1, S_2, \ldots S_n) = Op \oplus_k (S_1, Op \oplus_k (S_2, \ldots S_n))$$

L'**union** de deux séquences est un ensemble contenant les deux séquences

$$S \cup P = \{S, P\}$$

L'union de deux ensembles de séquences est l'union des ensembles.

$$\{S_1, S_2, \ldots, S_n\} \cup \{P_1, P_2, \ldots, P_m\} = \{S_1, S_2, \ldots, S_n, P_1, P_2, \ldots, P_m\}$$

La **fermeture de Kleene** d'un ensemble de séquences $\mathcal{S}$ représente l'ensemble de toutes les séquences que l'on peut bâtir par concaténations successives des séquences de $\mathcal{S}$. C'est la somme infinie de concaténations k-télescopées, i.e.,

$$\mathcal{S}* = \cup_{k=0}^{\infty} \mathcal{S}^k \tag{2.1}$$

avec

$$\mathcal{S}^{k+1} = \mathcal{S}^k \oplus_{k-1} \mathcal{S}^k \ , \ \mathcal{S}^0 = \epsilon \text{ et } S^1 = S$$

L'on remarquera que la fermeture de Kleene contient la chaine vide ϵ.

Tous les opérateurs sont n-aires. Cela signifie que l'expression $a \oplus (b \cup c) \oplus cd \oplus (a \cup c)$ sera traduite en $Op\oplus(a, (b \cup c), cd, (a \cup c))$. Un second exemple sera donné dans la figure 4.1. L'union est l'opérateur le moins prioritaire, suivie par la concaténation k-téléscopée et la fermeture de Kleene. L'union et la concaténation sont associatifs.

2.2 Caractérisation et types de contraintes

Les propriétés des contraintes sont des éléments déterminants pour l'extraction. Par exemple, l'anti-monotonie permet un élagage efficace de l'espace de recherche, et la diminution du nombre de séquences à considérer est exploité par la majorité des algorithmes. Les contraintes succinctes permettent d'énumérer directement les candidats sans nécessairement explorer tout l'espace de recherche en générant et testant toutes les séquences possibles.

- L'**anti-monotonie** signifie que si une séquence quelconque S' viole la contrainte C, alors toute sur-séquence S de S' la viole aussi. Formellement, C est anti-monotone, seulement si $\forall S' \subseteq S$ donne $\neg C(S') => \neg C(S)$. L'exemple le plus connu de contrainte anti-monotone est la contrainte de fréquence minimale. Si une séquence n'est pas fréquente, aucune de ses sur-séquences ne le sera. Ce type de contrainte permet des élagages dans l'espace de recherche.
- La **monotonie** est une propriété qui stipule que si un sous-séquence S' de S satisfait une contrainte C, alors S le satisfait aussi. Ce type de contrainte permet des générations de candidats efficaces. L'exemple classique est la présence d'un symbole. La négation d'une contrainte monotone est anti-monotone et vice versa.
- Une généralisation des contraintes anti-monotones et monotones conduit à l'introduction des contraintes **convertibles monotones et anti-monotones** par Pei et al. [93]. Certaines contraintes peuvent être rendues anti-monotones ou monotones après l'introduction d'un nouvel ordre entre les symboles de l'alphabet. C'est une technique utile pour l'extraction des *ensembles fréquents* qui ne pourra pas être utilisée pour l'extraction des motifs séquentiels pour lesquelles l'ordre spatio-temporel ne permet pas de réordonner les symboles. La propriété convertible (anti-)monotone ne sera plus étudiée dans cette dissertation.
- La propriété **préfixe monotone / anti-monotone** est plus générale que la monotonie, anti-monotonie classique. Une contrainte $\mathcal{C}$ est préfixe anti-monotone si $\forall X \in Sol(C, D, \Sigma^*)$ et $P \prec X$, alors $P \in Sol(C, D, \Sigma^*)$. On rappelle que $P \prec X$ dénote la relation P est un préfixe de X. La préfixe monotonie est définie par : si $\forall P \in Sol(C, D, \Sigma^*)$ et $P \prec X$, alors $X \in Sol(C, D, \Sigma^*)$.
  Toutes les contraintes (anti-)monotones sont préfixe (anti-)monotones. La réciproque est fausse. Cette famille de contraintes peut être gérée de façon très efficace par les techniques de projection présentées dans la section 2.3.2.
- Les contraintes **succinctes** permettent d'énumérer exactement l'ensemble des candidats sans devoir tout générer et tester. Une condition nécessaire pour qu'une contrainte soit succincte est l'existence d'une relation d'ordre entre les séquences satisfaisant la contrainte qui permet leur énumération directe, sans que toutes les séquences possibles sur Σ soient générées est testées. Les contraintes succinctes ont été proposés par Ng et al. [84], et ensuite [60] pour les ensembles fréquents, mais [91]

mentionne des adaptations pour les séquences. Une définition pourrait être :

Définition 1. *Rappelons que* $Sol(\mathcal{C}, DB, \Sigma) = \{S \in \Sigma^* \mid \mathcal{C}(DB, S) = true\}$.

- $\mathcal{E} = \{S_1, S_2 \ldots\} \subseteq \Sigma^*$ *est un* ***ensemble succinct*** *s'il peut être représenté par un prédicat de sélection* $\sigma_P(S)$.
- $\mathcal{LS} \subseteq \Sigma^*$ *est un* ***langage succinct****, s'il existe une collection finie d'ensembles succincts* $\{\mathcal{E}_1, \mathcal{E}_1, \ldots \mathcal{E}_n\}$ *tel que* $\mathcal{LS}$ *s'exprime par une expression de* $\{\mathcal{E}_1, \mathcal{E}_1, \ldots \mathcal{E}_n\}$ *utilisant les opérateurs union et différence. Mathématiquement,* $\mathcal{LS}$ *est l'algèbre des ensembles succincts* $\{\mathcal{E}_1, \mathcal{E}_1, \ldots \mathcal{E}_n\}$.
- $\mathcal{C}$ *est une* ***contrainte succincte*** *seulement si* $Sol(\mathcal{C}, DB, \Sigma)$ *est un langage succincte.*

Par exemple, soit $\Sigma = \{a = 1, b = 2, c = 3, d = 4, e = 5\}$, $\mathcal{C}_1(S) = a \in S \wedge \|S\| < 5$, $\mathcal{C}_2(S) = sum(S) < 15$, et $\mathcal{C} = \mathcal{C}_1 \wedge \mathcal{C}_2$. $\mathcal{C}$ est succincte, car la solution $Sol(\mathcal{C}, DB, \Sigma) = \mathcal{E}(\mathcal{C}_1) - \mathcal{E}(\neg\mathcal{C}_2)$. $\mathcal{E}(\mathcal{C})$ est l'ensemble succinct généré par la contrainte $\mathcal{C}$.

Pousser les contraintes en la présence de plusieurs contraintes de natures différentes est très difficile, surtout en l'absence de contraintes anti-monotones ou monotones. Il existe donc des contraintes qui ne peuvent pas être intégrées directement dans l'extraction. On essaie, dans ces cas, de les relaxer et filtrer a posteriori le résultat. La difficulté de pousser les contraintes non anti-monotones à l'intérieur de l'extraction provient du fait qu'elles ne permettent pas l'élagage des candidats basé sur la fréquence (ou tout autre contrainte anti-monotone) dans les phases précoces de l'extraction. Cela conduit à l'explosion du nombre des candidats à un niveau supérieur. Ainsi, pour certaines contraintes non anti-monotones comme, par exemple, la plupart des contraintes d'expressions régulières [44], il arrive qu'une sous-séquence ne satisfasse pas la contrainte C, alors que la séquence entière la vérifie. Dans ce cas, l'élagage à base de la contrainte C supprimera le candidat, sa fréquence ne sera pas comptée, et de nombreuses sur-séquences apparaîtront de manière explosive quelques générations plus tard [25, 6].

2.2.1 Types de contraintes

Soit S une séquence d'épisodes, nous distinguerons divers types de contraintes.

- **fréquence**. Soit $D = \{T_1, T_2 \ldots T_n\}$ la base de données contenant les séquences. La contrainte de fréquence demande que le support (absolu) de tout motif S satisfaisant la contrainte soit supérieure au seuil $minSup$.

$$support(S) \geq minSup$$

- **élément**. On peut exiger/refuser la présence/absence d'un élément.

$$\psi\ i \in \{1 \ldots \|S\|\} \text{ tel que } S_i\ \theta\ V$$

  Ici $\psi \in \{\exists, \forall\}$ est un quantificateur universel, V est un épisode, et $\theta \in \{\subseteq, \neg \subseteq\}$ est un opérateur ensembliste.
- **longueur**. On peut imposer la longueur minimale/maximale des séquences.

$$\|S\|\ \theta\ n$$

  où $\theta \in \{\leq, \geq\}$ est un opérateur relationnel, et n un nombre entier.

– **motif**. On recherche les séquences contenant certaines sous-séquences $\{M_1, M_2 \ldots\}$ spécifiées par l'utilisateur.

$$\exists M \in \{M_1, M_2 \ldots\} \subset \Sigma^* \text{ tel que } M \subseteq S$$

– **agrégat**. Les contraintes agrégat agissent sur l'ensemble des symboles composant le motif séquentiel. Par exemple, on peut souhaiter que la somme, le maximum/minumum ou la moyenne des éléments soit supérieur/inférieur à un seuil.

$$func(S)\ \theta\ r$$

où $\theta \in \{\leq, \geq\}$ est un opérateur relationnel, r un nombre réel, et la fonction **func** $\in$ {**min, max, sum, avg**}.

– **expression régulière**. Soit E une expression régulière définie par la grammaire du tableau 4.1. La contrainte s'écrit $S \in \mathcal{L}(E)$, avec $\mathcal{L}(E)$ le langage généré par l'expression régulière E.

– **durée**. Si les éléments des séquences de la base sont munis d'un attribut caractérisant leur localisation spatio-temporelle, il est possible de contraindre la différence maximale/minimale entre la position des premiers et des derniers éléments des séquences extraites. Généralement, la durée est définie comme :

$$dur(S) = localisation(S_{\|S\|}) - localisation(S_1)$$

mais d'autres métriques sont également possibles. La contrainte de durée s'écrit

$$dur(S)\ \theta\ r$$

où $\theta \in \{\leq, \geq\}$ est un opérateur relationnel et r un nombre réel.

– **gap**. La distance (gap) entre les éléments successifs d'une séquence "creuse" peut être vue comme une métrique spécifique, plus complexe que la durée.

$$\forall j \in \{2 \ldots \|S\|\}\ ,\ localisation(S_j) - localisation(S_{j-1})\ \theta\ r$$

où $\theta \in \{\leq, \geq\}$ est un opérateur relationnel et r un nombre réel.

Le tableau 2.1 résumé les propriétés des contraintes répertoriées dans ce chapitre.

Contrainte	Cons	Anti	Mono	Succ
Fréquence	$\exists\{D_1, D_2, \dots D_n\} \subset DB$ tel que $\forall j \in \{1 \dots n\}, S \subseteq D_j$ et $m \geq r$	Oui	Non	Non
Élément	$\forall i \in \{1 \dots \|S\|\}, S_i \theta V$, $(\theta \in \{\subseteq, \supseteq\})$	Oui	Non	Oui
Élément	$\forall i \in \{1 \dots \|S\|\}, S_i \cap V = \phi$	Oui	Non	Oui
Élément	$\exists i \in \{1 \dots \|S\|\}, S_i \theta V$, $(\theta \in \{\subseteq, \supseteq\})$	Non	Oui	Oui
Élément	$\exists i \in \{1 \dots \|S\|\}, S_i \cap V = \phi$	Non	Oui	Oui
Longueur	$\|S\| \leq n$	Oui	Non	Oui
Longueur	$\|S\| \geq n$	Non	Oui	Oui
Sous-motif	$\exists M \in \{M_1, M_2 \dots\}$ tel que $M \subseteq S$	Non	Oui	Oui
Agrégat simple	$\max(S) \leq r$, $\min(S) \geq r$	Oui	Non	Oui
Agregat simple	$\max(S) \geq r$, $\min(S) \leq r$	Non	Oui	Oui
Agrégat simple	$\sum(S) \leq r$, valeurs positives	Oui	Non	Non
Agrégat simple	$\sum(S) \geq r$, valeurs positives	Non	Oui	Non
Agrégat difficile	$\sum(S) \theta r$, $(\theta \in \{\leq, \geq\})$ valeurs positives et negatives	Non	Non	Non
Agrégat difficile	$avg(S) \theta r$, $(\theta \in \{\leq, \geq\})$	Non	Non	Non
Expression régulière	$S \in \mathcal{L}(ER)$	Non	Non	Non
Durée	$Dur(S) \leq r$	Oui	Non	Non
Durée	$Dur(S) \geq r$	Non	Oui	Non
Gap	$Gap(S) \theta r$, $(\theta \in \{\leq, \geq\})$	Non	Non	Non

TAB. 2.1 – *Caractérisation des principales contraintes. Ce tableau est adapté de [92]. S est une séquence d'épisodes. n est un entier et r est un réel. V est un épisode, i.e., un ensemble de symboles de Σ : $V = \{v_1, v_2, \dots, v_{m(i)}\}$, avec $\forall j \in \{1 \dots m(i)\}, v_j \in \Sigma$. Les trois dernières colonnes précisent si une contrainte possède la propriété anti-monotone, monotone ou succincte.*

2.3 Algorithmes d'extraction de séquences

L'énoncé du problème. Etant donnée une base de séquences $DB = \{S_1, S_2 \dots S_w\}$ sur un alphabet Σ, et une conjonction de contraintes $C = C_1 \wedge C_2 \wedge \dots C_n$ et une contrainte de fréquence $freq(S) > minsup$ trouver toutes les séquences $S \in \Sigma^*$ pour lesquelles
- $freq(S) \geq minsup$
- $C(S) = true$

L'ensemble des séquences satisfaisant la contrainte $\mathcal{C}$ sera noté $Sol(\mathcal{C}, DB, \Sigma)$.

2.3.1 Famille des algorithmes Apriori

Après le développement de l'algorithme Apriori [2] pour les ensembles fréquents (itemset), Agrawal et Srikant ont étudié l'application du même principe à l'extraction de séquences fréquentes. Ce problème est plus complexe, car l'ordre des éléments devient important et plusieurs optimisations identifiées pour traiter des ensembles fréquents ne se transposent pas facilement aux séquences (voir, par exemple, le cas typique des représentations condensées).

Fidèles aux termes et usages généraux, utilisé dans l'extraction par niveaux des ensembles fréquents [71], nous appelons famille Apriori le même algorithme adapté aux séquences. Afin de réduire l'espace de recherche, ces méthodes exploitent la propriété d'anti-monotonie de la contrainte de fréquence minimale : aucune sur-motif d'une séquence infréquente ne pourra pas être fréquente. La famille Apriori parcourt l'espace de recherche par niveau (breadth first) et trouve l'ensemble de toutes les séquences fréquentes. Les principaux représentants de la famille Apriori pour l'extraction de séquences sont [3, 97, 71, 75]. La table 2.2 présente une abstraction de la méthode Apriori.

	Données en entrée : DB, minsup
nr.	Produit la liste des séquences fréquentes
1	$F_1 = \{s \in \Sigma \mid support(s) > minsup\}$
2	k = 1
3	while $F_k \neq \emptyset$
4	$\quad C_{k+1} = Gen\_Cand(F_k)$
5	$\quad Compter\_Supports(C_{k+1})$
6	$\quad F_{k+1} = \{S \in C_{k+1}$ telle que $support(S) \geq minsup\}$
7	$\quad$ k++
8	return $F_1 \cup F_2 \dots \cup F_k$

TAB. 2.2 – *Pseudo code de l'algorithme Apriori. Le comptage des support (ligne 5) se fait par le biais des arbres de hashage, comme sur la figure 2.1. La génération des candidats (ligne 4) au niveau k avec Gen_Cand est une concaténation (k-1)-téléscopée, i.e., Gen_Cand(F_k) retourne $F_k \oplus_{k-1} F_k$.*

Si l'on analyse cet algorithme, trois phases sont à distinguer. La première (lignes 1-2) permet d'amorcer l'algorithme. Lors de cette phase, une passe complète est effectuée sur la base de séquences afin de comptabiliser le support de tous les éléments s de Σ. Les items dont le support est supérieur à *minSup* seront retenus pour former l'ensemble F_1, c'est-à-dire l'ensemble des items(symboles) fréquents. Ce n'est qu'à partir de là que l'on peut

alors envisager l'itération basée sur la succession des phases de génération des candidats (ligne 4) et de comptage des supports (ligne 5) dont voici le détail :

- GEN_CAND : cette étape permet la création des motifs candidats de taille $k+1$ (C_{k+1}) à partir de l'ensemble des motifs fréquents de taille k (F_k). Un exemple de ce type de génération est détaillé ci-après lors de la présentation de l'algorithme GSP (Generalized Sequential Patterns).
- COMPTER_SUPPORTS : cette étape permet d'établir le support des motifs candidats. Pour cela, le processus va effectuer une passe sur la base (on parlera de passe DB) et va incrémenter les supports de tous les motifs candidats contenus dans C_{k+1} (chaque fois que le processus détermine qu'une séquence contient un motif candidat, le support de celui-ci est incrémenté).

Par la suite, le nouvel ensemble de motifs fréquents F_{k+1} est obtenu par sélection des motifs de C_{k+1} dont le support est supérieur ou égal à *minsup* (ligne 6). Le processus s'arrête lorsque F_{k+1} est vide (il n'est plus possible de générer de candidats, ligne 3). L'union des F_k constitue alors F, l'ensemble des motifs fréquents. C'est cet ensemble qui est donné comme résultat, (ligne 8).

GSP

Generalized Sequential Patterns (GSP) est publié en 1995 par Agrawal et Srikant [3]. Le code de GSP est une instanciation de l'algorithme générique 2.2. La génération des candidats se fait de la façon suivante : deux motifs z_1 et z_2 de longueur k peuvent générer un motif de longueur $k+1$ si lorsque l'on retire le premier item à z_1, on obtient un motif identique à z_2 privé de son dernier item. Autrement dit, z_1 est étendu à l'aide du suffixe de z_2. Le motif z généré est alors tel que l'item ajouté à z_1 est un événement à part entière (événement composé d'un seul item) s'il constituait un événement également à part entière dans z_2 ; sinon l'item est ajouté comme dernier item au dernier événement de z_1. Ainsi *cad* et *ada* permettront de générer le motif *cada* alors que *cad* et *ade* permettront de générer le motif *cade*. En ce qui concerne le cas particulier de la génération des motifs de taille 2, cela se fait par simple produit cartesien.

Un second article [97] développe une approche efficace pour optimiser le comptage des candidats. Il s'agit d'un arbre de hashage (hash tree) qui contient la collection des séquences candidates. Chaque chemin de l'arbre correspond à une succession des clefs de hashage, et les feuilles contiennent la collection des candidats (un compteur est associé à chaque candidat).

Un exemple d'arbre de hashage est donné par la figure 2.1. Dans cet arbre, un noeud peut être soit une table de hashage dont chaque entrée pointe vers un noeud de niveau immédiatement supérieur (noeuds *intérieurs*), soit une liste de motifs (*feuilles*).

Pour **ajouter** un motif candidat dans l'arbre de hashage, GSP parcourt l'arbre, appliquant à chaque niveau k (le niveau de la racine est 1) une fonction de hashage sur le $k$ième item du motif, et ce de façon à atteindre une feuille dans laquelle il puisse insérer le motif. Si l'ajout dans cette feuille du motif porte le nombre de motifs contenus dans la feuille à u niveau supérieur à une limite spécifiée au préalable, alors la feuille est scindée en deux ou plusieurs feuilles, l'ensemble de celles-ci reprenant l'intégralité des motifs contenus dans

la feuille mère. Quant à celle-ci, elle est transformée en nœud intérieur pointant sur les feuilles nouvellement générées.

Lors du **comptage**, toutes les séquences de la base de séquences sont projetées dans cette arbre de hashage. En cas de "match", le compteur du candidat correspondant est incrémenté. Cette technique est très performante. Le candidats sont identifiés facilement, le nombre de comparaisons symbole par symbole des séquences de la base de données est minimal. L'exemple de la figure 2.1 montre un arbre de hashage représentant les candidats : *abb, abc, aec, bce, ebc, ebd, eca et ace.*

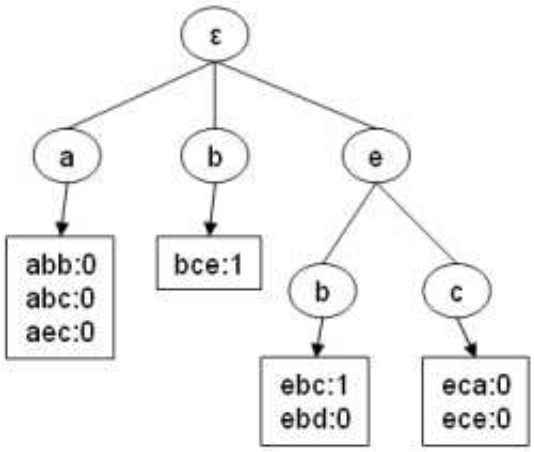


FIG. 2.1 – *Arbre de hashage de GSP. Pour insérer la séquence S, nous avons utilisé une fonction de hashage correspondant le k^{eme} caractère de S au niveau k. Un niveau supplémentaire est crée si une liste contient plus de trois éléments. Les compteurs correspondent à l'état après l'examen de la transaction de $T = aebce$.*

Par exemple, lors du comptage des candidats au niveau trois, toutes les 3-séquences de la séquence *aebce* sont projetés dans l'arbre de hashage de la figure 2.1, notamment : *aeb, ebc, bce. ebc* et *bce* correspondent à des candidats, dont GSP incrémentera les compteurs respectives. *aeb* n'es pas un candidat dans l'arbre de hashage et sa projection dans l'arbre de hashage ne produira aucun effet.

PSP

Masseglia et ses co-auteurs proposent dans [75] d'utiliser un arbre préfixe pour stocker les candidats. Dans cette structure, chaque chemin de l'arbre correspond à une séquence unique, les branches étant les transitions. La figure 2.2 reprend l'arbre de hashage de la figure 2.1 et en fait un arbre préfixe.

S'appuyant sur cette structure, les auteurs construisent l'algorithme Prefixe Tree for Sequential Patterns (PSP) qui dispose de bonnes propriétés :

- consommation en mémoire réduite, car l'arbre sert a stocker directement les candidats, alors que les arbres de hashage sont pénalisés par le stockage d'une structure de clefs d'accès en sus des candidats
- identification plus rapide des séquences correspondant à des candidats. Une fois arrivé jusqu' à la feuille, GSP doit comparer la séquence à tous les candidats dans

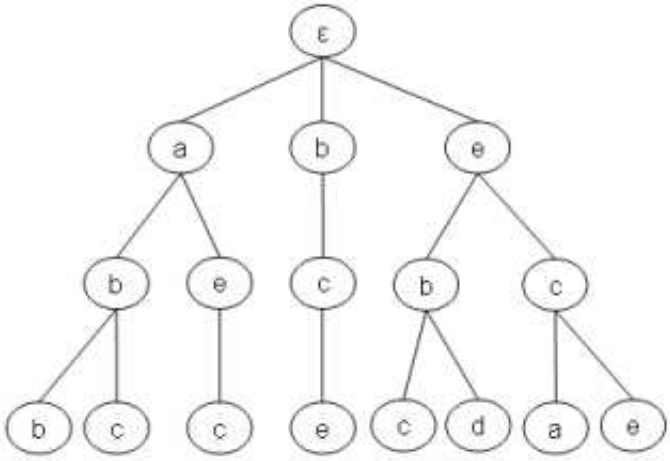


FIG. 2.2 – *Arbre préfixe de GSP (même collection de séquences que figure 2.1)*

la feuille, alors que pour PSP chaque feuille correspond à un candidat. En même temps, l'arbre de hashage est moins profond ... donc la différence de performance dépend beaucoup du calibrage de l'arbre de hashage et des données.

Winepi, Minepi

Dans [71], Mannila et al. recherchent les épisodes qui apparaissent suffisamment souvent dans une séquence d'épisodes. Au lieu d'avoir une base de données contenant une collection de séquences, les données apparaissent ici comme une seule série temporelle. Cette structure de données est typique, par exemple, aux cours de bourse. Comme nous ne nous intéressons qu'aux séquences d'évènements simples (deux évènements ne peuvent pas avoir lieu au même moment), nous ne présenterons pas les définitions supplémentaires liées aux épisodes. Ainsi, Minepi et Winepi seront simplifiés dans notre présentation dans la table 2.3.

	Données en entrée : winSize, minsup, séquence R
nr.	Produit la liste des séquences fréquentes dans R
1	$C_1 = \{S \in \Sigma\ \}$
2	k = 1
3	while $C_k \neq \emptyset$
4	$\quad F_k = \{S \in C_k \mid support(S, R, winSize) > minsup\}$
5	$\quad$ k++
6	$\quad C_k = \{S \in \Sigma * \mid \|S\| = k \wedge (\forall P \subset S \Rightarrow P \in F_{\|P\|})\}$
7	return $F_1 \cup F_2 \ldots \cup F_k$

TAB. 2.3 – *Pseudo code de Winepi/Minepi. L'algorithme a été simplifié pour ne traiter que des séquences de symboles. Les deux algorithmes diffèrent seulement dans leur définition du support.*

Le $support(S, R, winSize)$ de la séquence d'épisodes S dans la méthode Winepi est le nombre d'occurrences de S en R. L'*occurrence* de S dans la séquence d'épisodes R est une *fenêtre glissante* de longueur $winSize$ contenant au moins une fois l'épisode S.

L'algorithme Minepi repose sur une définition d'occurrence différente. Maintenant, l'occurrence d'un épisode $\mathcal{E}$ n'est pas une fenêtre f quelconque de taille $winSize$ contenant $\mathcal{E}$. En effet, pour Minepi la fenêtre f ne peut pas contenir une autre fenêtre f' dans laquelle on trouverait le même épisode.

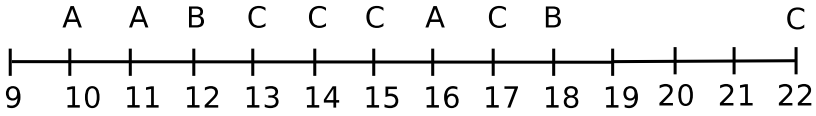


FIG. 2.3 – *Exemple de données pour Winepi/Minepi.*

Sur la figure 2.3, en prenant la taille de fenêtre égale à 3, le support de la séquence AB vaut 2 avec Minepi, et 3 avec Winepi.

2.3.2 Techniques de projection

L'équipe de l'Université Simon Fraser a proposé de partitionner la base de données initiale en fonction des préfixes des séquences. Cela permet une exécution parallèle et dirige l'effort d'extraction vers la partie utile de la base de données, i.e., celle qui peut contenir des motifs solutions. Par exemple, si on ne souhaite connaitre que les séquences qui commencent par le préfixe ab, ces méthodes construiront la projection de la base initiale sur le préfixe ab, réduisant ainsi l'effort d'extraction en évitant de considérer la partie de la base qui n'est pas préfixée par ab. L'économie réalisée peut être conséquente. Les méthodes de projection ne génèrent pas de candidats stricto sensu.

Définition 2. *Soient S et P deux séquences de Σ^*, tel que $P \subseteq S$. On appelle* ***projection*** *de S sur P la séquence R satisfaisant les propriétés suivantes :*

- $R \triangleleft S$
- $P \prec R$
- *R est maximale, i.e.,* $\forall (Q \triangleleft S*) \wedge (R \subset Q) \Rightarrow P \not\prec Q$

La projection d'une base de données D sur une séquence S est une nouvelle base de données $D_{|S}$ contenant la projection sur S de toutes les séquences de D.

La table 2.4 précise comment on calcule une projection. Par exemple, la projection de $afccbacae$ sur acb est $acbacae$.

PrefixSpan

Pei et al. ont proposé PrefixSpan [90] en 2001. C'est un algorithme qui identifie d'abord tous les k-préfixes fréquents, construit ensuite la base de données projetée sur chacun des k-préfixes fréquents et s'appelle de façon récursive sur chacune des projections. Ce principe est résumé dans la table 2.5. L'algorithme donné dans la table 2.5 a été modifié par rapport

nr.	Données en entrée : S, P Produit la projection de S sur P
1	k = 1
2	forall $i = 1 \dots \|P\|$
3	if $i > \|S\|$ return ϵ
4	if $P_i \neq S_k$: i++ ; goto 3
5	k++
6	return $P_1 P_2 \dots P_{\|P\|} S_k S_{k+1} \dots S_{\|S\|}$

TAB. 2.4 – *Calcul d'une projection pour les séquences de symboles (voir, e.g., [91] pour la projection de séquences d'épisodes).*

à la publication originale afin d'assurer que le préfixe corresponde à des symboles adjacents des séquences projetées.

nr.	PrefixSpan(DB, m, P) Extrait les séquences S satisfaisant $support(S) \geq m$ et P est préfixe de S
1	$k = \|P\|$
2	$Pr \leftarrow (k+1)$-préfixes de DB (une passe sur DB)
3	purge DB des éléments infréquents ou inutiles
4	forall $R \in Pr$
5	print R
6	calcule $DB_{\|R}$ en utilisant la méthode du listing 2.4
7	call PrefixSpan($DB_{\|R}$, m, R)

TAB. 2.5 – *Pseudo-code de PrefixSpan. Appel initial PrefixSpan(DB, minsup, ϵ)*

Suivons l'exécution de PrefixSpan sur les figures 2.4 à 2.7. Cette figure donne également les premières projections réalisées par PrefixSpan sur toutes les séquences 1-préfixes fréquentes. On remarquera la disparition des symboles infréquents. Ceci est représenté par le symbole _. Par exemple, la projection de *acdbdf* sur *c* sera notée *_cd_d_*, car ni *b* ni *f* ne sont fréquents. La première étape de l'exécution calcule la fréquence des 1-préfixes : *a :6, b :2, c :5, d :5, e :1, f :1.* *a*, *c* et *d* sont les seules séquences fréquentes, PrefixSpan enlèvera donc *b* ,*e*, et *f* de la base et calculera les projections sur *a*, *c* et *d*.

Traitons d'abord l'exploration de la projection sur les préfixes commençant par *a*. Dans la projection sur *a*, l'algorithme trouvera les préfixes *ac* : 5 et *ad* : 2. Le préfixe *ac* apparait 5 fois, il faut donc calculer la projection sur *ac*. Dans cette projection, le 3-préfixe *acd* apparait 3 fois, il fait donc partie de la solution au même titre que *a* et *ac*. Enfin, la projection sur *acd* est vide. Ce processus est illustré par la figure 2.5.

L'exploration de la projection sur *c* contient la séquence 2-préfixe fréquente *cd*. La base projetée sur *cd* est vide, ce qui termine l'exploration des séquences commençant par *c*.

La dernière partition contient les séquences préfixées par *d* : 5. Au niveau 2, *da* : 4 est fréquente, ce qui déclenche une projection sur *da*. *dac* : 3 est fréquente (et *dad* : 1 ne l'est pas), mais sa base projetée ne contient aucun 4-préfixe dont la fréquence dépasserait *minsup=3*, i.e. *dacd* : 2.

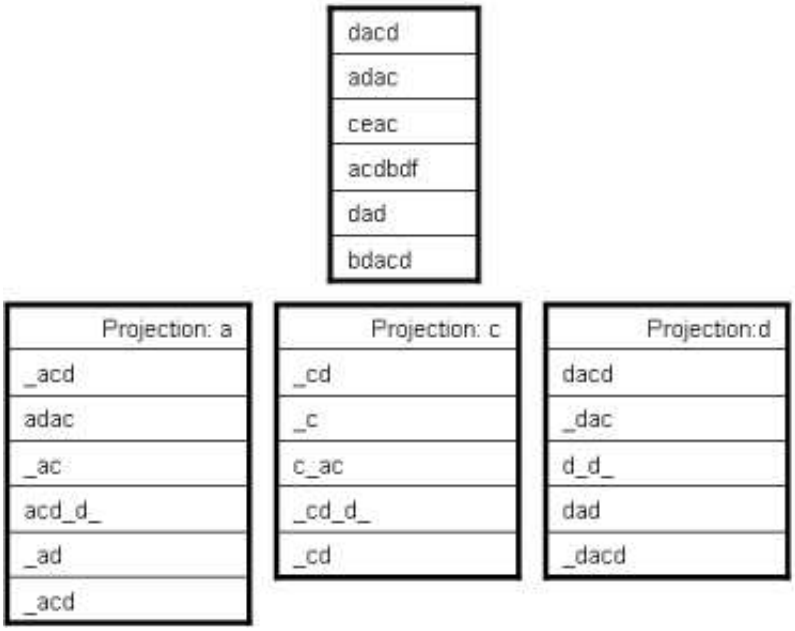


FIG. 2.4 – *Exemple d'exécution de PrefixSpan. Étape 1 : projections sur a, b, c.*

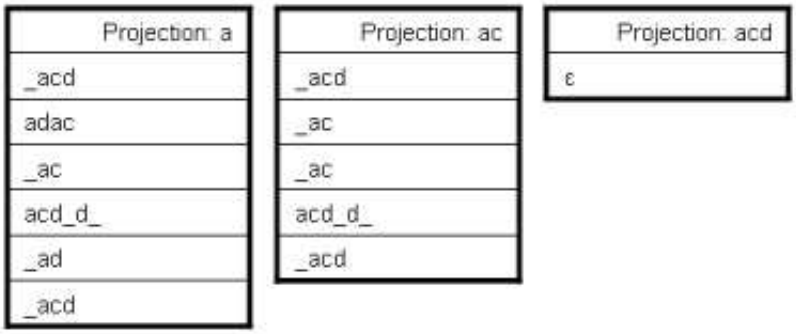


FIG. 2.5 – *Exemple d'exécution de PrefixSpan. Étape 2 : projections sur a, ac et acd.*

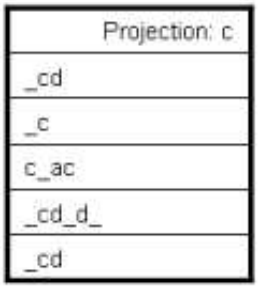


FIG. 2.6 – *Exemple d'exécution de PrefixSpan. Étape 3 : projections sur c et cd.*

FIG. 2.7 – *Exemple d'exécution de PrefixSpan. Étape 4 : projections sur d, da dac et dacd.*

L'extraction est achevée. PrefixSpan a trouvé les séquences : *a :6, ac :5, acd :3, c :5, cd :3, d :5, da :4, dac :3.*

Pattern-growth avec contraintes

Les techniques de projection permettent de gérer facilement les contraintes préfixe (anti)monotones. En effet, il suffit de restreindre les préfixes à celles satisfaisant la contrainte pour réaliser un élagage puissant.

Les contraintes qui n'ont pas la propriété préfixe (anti-)monotone, comme certains agrégats (présence d'éléments positifs et négatifs, moyenne), peuvent être gérées par une variante ad-hoc de la méthode générale. Le lecteur intéressé pourra se référer à [91]. L'algorithme *Pattern-growth avec contraintes* résumé dans la table 2.6 est très proche de PrefixSpan (table 2.5).

	constrainedPatternGrowth(DB, $\mathcal{C}$, P)
nr.	Extrait les séquences S satisfaisant $\mathcal{C}$ et P est préfixe de S
1	$k = \|P\|$
2	$Pr \leftarrow (k+1)$-préfixes de DB (une passe sur DB)
3	purge DB des éléments infréquents ou inutiles
4	forall $R \in Pr$ such as $\mathcal{C}(R) = true$
5	print R
6	calcule $DB_{\|R}$ en utilisant la méthode de la table 2.4
7	call constrainedPatternGrowth($DB_{\|R}$, $\mathcal{C}$, R)

TAB. 2.6 – *Pseudo-code de PrefixSpan pour les contraintes préfixe anti-monotones. L'extraction est déclenchée par constrainedPatternGrowth(DB, $\mathcal{C}$, ϵ). $\mathcal{C}$ est une conjonction de contraintes préfixe anti-monotones.*

Pour traiter des contraintes préfixes monotones, l'algorithme de la table 2.6 doit être modifié :

4 forall $R \in Pr$ such as **potentially** $\mathcal{C}(R) = true$
5 **if** $\mathcal{C}(R) = true$, print R

Les autres lignes restent inchangées.

Il est intéressant de noter qu'une contrainte d'expression régulière peut être relaxée

pour devenir une contrainte préfixe anti-monotone. La contrainte relaxée s'écrit $S \in trunc(\mathcal{L}(ER), \|S\|)$ où la fonction $trunc(\mathcal{L}(ER), n)$ représente la troncature du langage généré par ER après n symboles.

Afin d'illustrer les techniques de projection avec des contraintes, attribuons des valeurs aux symboles de Σ : a=1, b=2, c=3, d=4, e=15, f=8. Soit la contrainte anti-monotone $sum(S) < 5$, et reprenons l'exemple des figures 2.4 - 2.7. Les préfixes *ad, cd, da* ne satisfont pas la contrainte et les projections respectives ne seront pas calculées. Donc, au lieu de calculer 8 projections comme dans le cas non-contraint, la nouvelle méthode calculera seulement 4 projections et trouvera la solution : *a :6, ac :5, c :5, d :5.*

CloSpan

Le nombre de motifs fréquents renvoyés par un extracteur est souvent gigentesque, et il y a eu des efforts pour trouver des représentations condensées au sens de [28, 29]. Ainsi, les qualités des ensembles fermés sont désormais bien connues et de nombreux algorithmes exploitent ces notions pour calculer des ensembles fréquents dans des cas de données denses et fortement corrélées (voir, par exemple, [88, 108, 89]). Constatant les gains de vitesse et la réduction de l'espace de stockage des motifs trouvés, les chercheurs ont tenté de transposer ces notions aux séquences. Malgré quelques tentatives (voir, e.g., [105]), la définition de ce que deviennent les fermetures pour les séquences n'est toujours pas clairement établie.

2.3.3 Techniques basées sur listes d'occurrences

Les approches de type Apriori sont très consommatrices en accès aux disques. En effet, chaque phase de comptage, aussi optimisée soit-elle, déclenche une lecture de toute la base de séquences. Une solution consiste alors à stocker les informations de la base de séquences en mémoire vive, sous la forme de *listes d'occurrences*.

Cette famille de méthodes [106, 107, 15, 65] est basée sur une représentation de la base de séquences par des liste d'occurrences des événements. La génération des candidats est une opération de jointure de listes, le comptage des fréquences consiste à regarder la cardinalité des jointures. Par leur construction, les algorithmes sur listes d'occurrences gèrent de façon efficace la majorité des contraintes spatio-temporelles : maxGap, minGap, maxWin, présence ou absence d'un événement, etc. L'**occurrence** d'un évènement est un triplet *(SID, début, fin)*, où *SID* est l'identifiant (numéro) de la séquence contenant l'événement, alors que *début* et *fin* contiennent l'information de localisation spatio-temporelle. Par exemple, la séquence *abbc* serait représentée comme a(1,1,1), b(1,2,2), b(1,3,3), c(1,4,4), et sa sous-séquence généralisée *ac* serait notée *ac(1,1,4)*.

Spade

L'algorithme Spade [107] , sera présenté dans une spécification quelque peu réduite comparée à la communication initiale de M. Zaki. En effet, ce mémoire ne s'intéresse qu'aux séquences d'évènements simples. Le concept de base de Spade est la liste d'occurrence d'une séquence (ou motif), appelé IdList. La figure 2.8 montre comment Spade représente une base de séquences par des listes d'occurrence. Curieusement, les listes d'occurrences permettent de généraliser la notion de séquence et d'extraire des *séquences creuses*, i.e., avec une distance spatio-temorelle entre les évènements successifs.

SID	séquence
1	abbc
2	cab
3	acb
4	caba
5	bada

LISTID = a	
SID	début=fin
1	1
2	2
3	1
4	2
4	4
5	5
5	4

LISTID = b	
SID	début=fin
1	2
1	3
2	3
3	3
4	3
5	1
5	3

LISTID = c	
SID	début=fin
1	4
2	1
3	2
4	1

LISTID = d	
SID	début=fin
5	3

FIG. 2.8 – *Spade représente la base de séquences sous la forme d'une liste par évènement*

La génération des candidats est effectuée par une jointure temporelle (à base de précédence dans une fenêtre de temps donnée ou de l'occurrence simultanée) des séquences fréquentes projetées en classes d'équivalence. Il s'agit d'un algorithme de type *divide et impera (diviser pour régner)*. L'espace de recherche est subdivisé en classes d'équivalence préfixes M_i. La classe M_i contient tous les motifs contraints ayant le motif M_i pour préfixe. L'intérêt de cette approche est qu'elle permet la parallélisation de l'algorithme, chaque classe d'équivalence pouvant être traitée de façon indépendante.

L'algorithme Spade introduit deux types d'opérations :

- la **fusion** des deux séquences qui partagent le même préfixe permet de construire des séquences plus longues. Formellement, si deux séquences $S_1 = Px$ et $S_2 = Py$ partagent le même préfixe P, alors *fusion* (S_1, S_2) = Pxy.
- la **jointure temporelle** des IdList des séquences S_1 et S_2 est une nouvelle IdList, i.e., $joinT(S_1, S_2) := IdList(fusion(S_1, S_2))$. Soit $Z = fusion(S_1, S_2)$. Alors, pour chaque triplet $(SID_1, d_1, f_1) \in IdList(S_1)$ et $(SID_2, d_2, f_2) \in IdList(S_2)$, satisfaisant $SID_1 = SID_2$ et $f_1 < d_2$, ajouter (SID_1, d_1, f_2) à $IdList(Z)$.

Le pseudo-code Spade donné dans la table 2.7 est intuitif. F_k est la liste de séquences creuses fréquentes de longueur k.

L'exemple des figures 2.9 et 2.11 montre l'exécution de Spade sur la base de séquence de la figure 2.8, avec la contrainte de fréquence $minsup = 2$. Les deux premières générations sont illustrées par les figures 2.9 et 2.10 - identification des 1 et 2-séquences.

Une troisième génération est donnée par la fusion des listes qui contiennent au moins

nr.	Données en entrée : DB, minsup Produit la liste des séquences fréquentes
	Lire DB pour calculer F_1, et les IdList(S) pour tout $S \in F_1$
1	k = 1
2	while $F_k \neq \emptyset$
3	{
4	$F_{k+1} = \emptyset$
5	forall $P \in F_k, Q \in F_k$ if prefixe(P)=prefixe(Q)
6	{
7	Z = fusion(P, Q)
8	Construct IdList(Z)
9	if freq(Z) > minsup $F_{k+1} = F_{k+1} \cup \{Z\}$
10	}
11	$k = k + 1$
12	}
13	return $F_1 \cup F_2 \ldots \cup F_k$

TAB. 2.7 – *Pseudo code de Spade.*

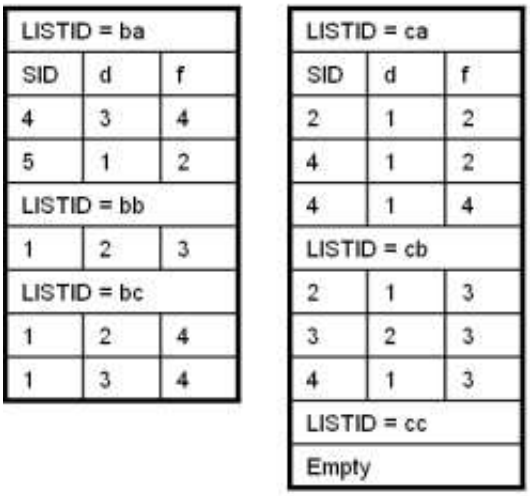
LISTID = aa		
SID	d	f
4	2	4
5	2	4
LISTID = ab		
1	1	2
1	1	3
2	2	3
3	1	3
4	2	3

LISTID = ac		
SID	d	f
1	1	4
3	1	2

FIG. 2.9 – *Jointures temporelles pour le préfixe a.*

deux éléments. La fusion des listes avec le préfixe *a* calcule les jointures temporelles de *aaa, aab, aac, aba, abb, abc, aca, acb.* Aucune n'est fréquente. La partition qui commence par *b* examine les jointures *baa, bac, bca, bcc.* L'opération ne génère aucune liste de plus d'un élément. Enfin, la dernière partition commençant en *c* calcule *caa, cab, cba, cbc.* Cela fournit la seule séquence fréquente de longueur 3. Le résultat est représenté sur la figure 2.11. La quatrième génération teste l'IdList de *cabb* qui est vide.

Globalement, Spade aura identifié les séquences d'évènements *a, b, c, aa, ab, ac, ba, bc, ca, cb,* et *cab.*



LISTID = ba		
SID	d	f
4	3	4
5	1	2
LISTID = bb		
1	2	3
LISTID = bc		
1	2	4
1	3	4

LISTID = ca		
SID	d	f
2	1	2
4	1	2
4	1	4
LISTID = cb		
2	1	3
3	2	3
4	1	3
LISTID = cc		
Empty		

FIG. 2.10 – *Jointures temporelles pour les préfixes b et c.*

LISTID = cab		
SID	d	f
2	1	3
4	1	3

FIG. 2.11 – *Jointures temporelles pour le préfixe cab*

cSpade

Développé par M. Zaki comme une extension de Spade [107], cSpade apporte la gestion de plusieurs contraintes [106] : longueur et largeur maximales des motifs, distances minimales exigées et distances maximales permises entre les motifs d'une séquence, taille de fenêtres, présence ou absence de motifs. L'ajout de ces contraintes est assez intuitif. Soit $(X, d_1, f_1) \in IdList(S_1)$ et $(X, d_2, f_2) \in IdList(S_2)$

- **Absence d'un événèment** L'évenement non désiré est effacé de F_1.
- **maxLen** La jointure temporelle vérifie que la $f_2 - d_1 < maxLen$.
- **minGap** La jointure temporelle vérifie que la $d_2 - d_1 > minGap$.
- **maxGap** La jointure temporelle vérifie que la $d_2 - f_1 < maxGap$.
- Tout autre contrainte **anti-monotone** se greffe facilement sur ce schéma.

Il faut faire attention lors de la gestion de la contrainte maxGap qui n'est pas anti-monotone. En effet, une séquence non valide peut générer une séquence plus longue valide. Si la contrainte est $maxGap < 3$ et les séquences $b(2)d(4)$ et $b(2)f(5)$ sont fréquente, le gap dans $b(2)f(5)$ vaut 3, ce qui est est supérieur à maxGap. Cependant, $b(2)d(4)f(5)$ est un candidat valide. Afin de palier à ce problème, M. Zaki modifie la génération des candidats. Au lieu de fusionner deux IdList de longueur k, il propose de fusionner F_k avec F_2, et affaiblit l'élagage à base de fréquence.

Go-Spade et GoSpec

M. Leleu [65, 66] a remarqué que l'efficacité des opérations de jointure dépend grandement de la représentation choisie pour l'occurrence d'un motif dans sa liste et que les données avec un grand nombre de répétitions consécutives (symboles répétés) mettaient les algorithmes Spade et cSapde en défaut. Elle a donc étudié une compression des listes d'occurrences avec la méthode suivante : si le dernier item d'un motif apparaît plusieurs fois dans une séquence de façon immédiate et consécutive sur un intervalle de temps donné, alors, plutôt que de noter chaque occurrence, on note l'intervalle de temps dont la date de départ est la date d'apparition du suffixe de la première occurrence, et dont la date de fin est la date d'apparition du suffixe de la dernière occurrence consécutive. Il a été montré dans [65] qu'une telle représentation permet bien de générer tous les motifs fréquents. Les améliorations en termes d'extraction de séquences fréquentes et d'extractions sous des formes plus générales de contraintes ont été présentées respectivement dans deux algorithmes [66] et [67].

Mentionnons également l'algorithme SPAM [15] comme une autre contribution aux algorithmes basés sur des listes d'occurrences. Il utilise des représentations booléennes des listes représentant les occurrences, ce qui permet de réduire la consommation en mémoire. Les jointures sont ramenées à des opérations binaires très efficaces.

2.3.4 Techniques basées sur les espaces des versions

L' **arbre suffixe** de la chaine S prise d'un alphabet Σ est une structure de données arborescente, dont les arrêtes sont marqués par les éléments de Σ, de manière a ce que la concaténation des marques sur une branche soit un suffixe de S. Toutes les feuilles d'un arbre suffixe représentent de manière unique un suffixe de S. La racine contient la chaine vide. La figure 2.12 représente l'union des suffixes de la collection $\{abd, bdab, adab, babda\}$.

Une fois l'arbre suffixe construit, les séquences fréquentes peuvent être identifiées par un simple parcours d'arbre. Cette technique est similaire à l'algorithme FP-Growth de Han et. al. [50, 51] connu pour le calcul des ensembles fréquents.

L' **espace des versions** des chaines est la structure $\mathcal{V}$ satisfaisant $\forall P, Q, R \in \Sigma^*$ tel que $P, R \in \mathcal{V}$ et $P \subseteq Q \subseteq R \Rightarrow Q \in \mathcal{V}$. Définissions l'ensemble spécifique S et l'ensemble générique G représentant la solution de l'extraction sous les contraintes $\mathcal{A} \wedge \mathcal{M}$ ($\mathcal{M}$ dénote une conjonction de conjonctions de contraintes monotones et $\mathcal{A}$ une conjonction de conjonctions de contraintes anti-monotone) par

- $S = \{p \in Sol(\mathcal{A} \wedge \mathcal{M}, D, \Sigma) \mid \forall q, \text{ tel que } q \subseteq p \Rightarrow q \notin Sol(\mathcal{A} \wedge \mathcal{M}, D, \Sigma)\}$
- $G = \{p \in Sol(\mathcal{A} \wedge \mathcal{M}, D, \Sigma) \mid \forall q \subseteq p \Rightarrow q \notin Sol(\mathcal{A} \wedge \mathcal{M}, D, \Sigma)\}$

Les ensembles S et G définissent de manière unique l'espace des versions $\mathcal{V}$. Mathematiquement, un espace des versions est u treillis représentant un ordre partiel. Par exemple, la relation binaire sousChaine(X,Y) définit un ordre partiel et se laisse représenter par un espace des versions.

Corolaire. L'espace des versions $\mathcal{V}$ représente la solution $Sol(\mathcal{A} \wedge \mathcal{M}, D, \Sigma)$ d'une extraction si l'ensemble spécifique et l'ensemble générique satisfont :

- $\forall p \in Sol(\mathcal{A} \wedge \mathcal{M}, D, \Sigma), \exists s \in S \wedge g \in G$ tel que $g \subseteq p \subseteq s$
- $\forall p, q, r \in \Sigma^*$, tel que $p \subseteq q \subseteq r, p, r \in Sol(\mathcal{A} \wedge \mathcal{M}, D, \Sigma) \Rightarrow q \in Sol(\mathcal{A} \wedge \mathcal{M}, D, \Sigma)$

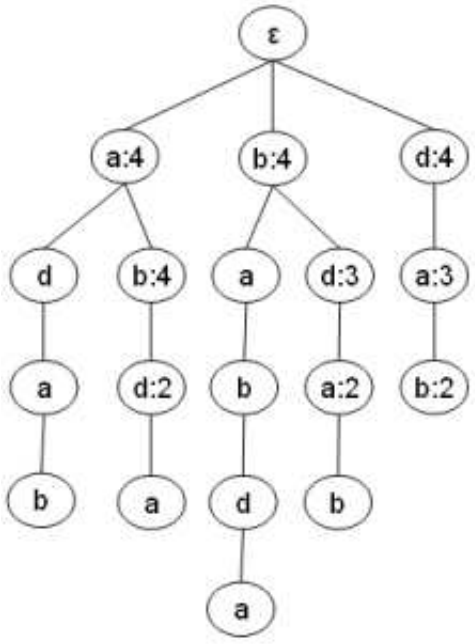


FIG. 2.12 – *Exemple d'arbre suffixe pour la base de séquences* $\{abd, bdab, adab, babda\}$.

Par exemple, dans le cas de la figure 2.12 avec minsup=2, $G = \{a, b, d\}$ et $S = \{abd, bda, dab\}$.

VST

De Raedt et son équipe traitent de manière efficace l'extraction sous contraintes monotones (notés $\mathcal{M}$) et anti-monotone (notés $\mathcal{A}$) des chaines de caractères avec l'algorithme VST (Version Space Trees) dans le cas où la solution peut être représentée par la classique espace des versions.

L' **arbre VST** sont une généralisation des arbres suffixes, dans laquelle les nœuds ont été décorés par les symboles $\oplus$ et $\ominus$ afin d'intégrer la gestion des contraintes. Chaque nœud de l'arbre VST suffixe contiendra le label $\oplus$, si la séquence représente par cette branche fait partie de la solution $Sol(\mathcal{A} \wedge \mathcal{M}, D, \Sigma)$, et $\ominus$ sinon. Dans cette structure l'ensemble spécifique est donné par l'union des feuilles du VST, alors que l'ensemble générique contient tous les nœuds marqués $\oplus$ dont le parent est marqué $\ominus$ on non-existant.

L'algorithme VST procède en deux étapes :

- DESCEND : Construction descendante de l'espace des versions satisfaisant la contrainte anti-monotone $\mathcal{A}$ utilisant Apriori.
- ASCEND : Vérification ascendante de la contrainte monotone $\mathcal{M}$ (donné par le tableau 2.8).

Exemple d'exécution. Suivons l'exécution de VST sur la base de chaines $DB =$ { *acca, ggtat, cga, accag , atga* } sous la conjonction de la contrainte anti-monotone $minsup \geq 2$ et monotone "contient la chaine *ca*".

Étape 1. Exécution d'apriori et représentation des chaines fréquentes sur la figure 2.13.

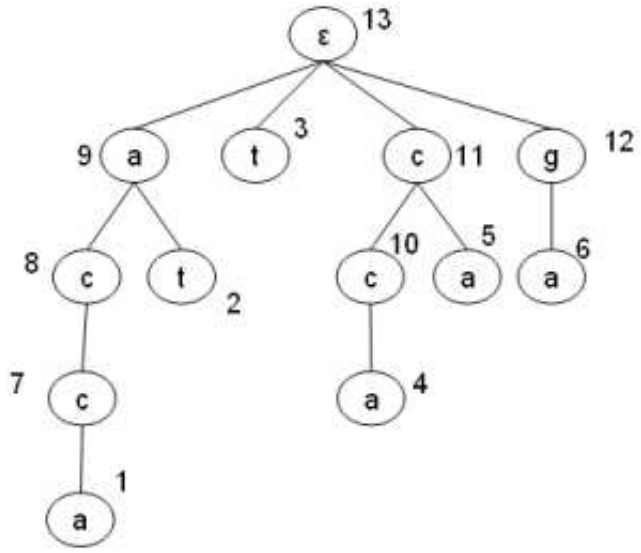


FIG. 2.13 – *Exemple d'exécution VST. La figure contient un arbre VST représentant la partie anti-monotone de la contrainte, i.e., minsup* ≥ 2. *Les nœuds ont été numérotés pour faciliter l'explication.*

Étape 2. Vérification de la contrainte monotone.
- $C_1 = \{1, 2, 3, 4, 5, 6\}$ - toutes les feuilles de l'arbre VST calculé par DESCEND.
- $F_1 = 1, 4, 5$, car $\{at,\ t,\ ga\ \}$ ne contient pas *ca*. $C_2 = \{7, 10, 11\}$. La figure 2.14 représente cette situation.

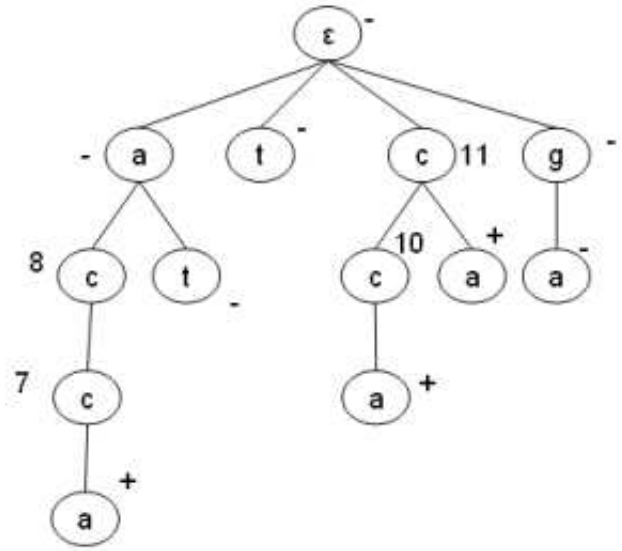


FIG. 2.14 – *Exemple d'exécution VST. ASCEND commence à décorer l'arbre par* $\oplus$ *et* $\ominus$.

– $F_2 = \{7, 10, 11\}$, $C_3 = \{8\}$
– et finalement $F_3 = \{8\}$, $C_4 = \emptyset$, ce qui arrête l'extraction. La VST représentant la conjonction des deux contraintes est sur la figure 2.15.

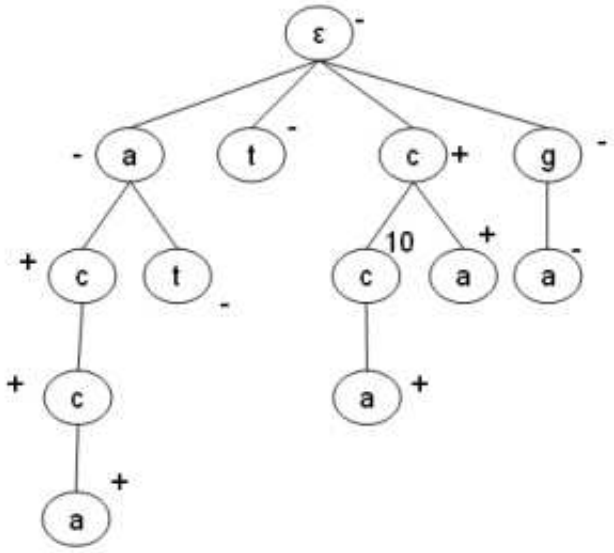


FIG. 2.15 – *Exemple d'exécution VST. L'arbre VST finale représentant la conjonction de minsup* ≥ 2 *et "contient ca". La solution se trouve dans les nœuds marqués* $\oplus$ *: acca, cca, ca.*

nr.	Données en entrée : $DB, \mathcal{V}^{\mathcal{A}}$, $\mathcal{M} = \wedge_{i=1}^{n} \mathcal{M}_i$ Produit un VST satisfaisant $\mathcal{A}$ et $\mathcal{M}$
1	$C_1 \leftarrow$ { feuilles de$\mathcal{V}^{\mathcal{A}}$}
2	k $\leftarrow 1$
3	tant que $C_k \neq \emptyset$ faire
4	{
5	$F_k = \{N \in C_k \mid N \in Sol(\mathcal{A} \wedge \mathcal{M}, D, \Sigma)\}$
6	$C_{k+1} = \emptyset$
7	pour tout $N \in F_k$ faire
8	marque N $\oplus$
9	$F_{k+1} = F_{k+1} \cup N.pere$
10	pour tout $N \in C_k - F_k$ faire
11	marque N $\ominus$
12	marque le chemin depuis N à la racine $\ominus$
13	k++
14	}
15	retourne $\mathcal{V}^{\mathcal{A}}$

TAB. 2.8 – *Pseudo code de VST :ASCEND.* $\mathcal{V}^{\mathcal{A}}$ *est l'arbre VST construit par DESCEND et représentant l'espace des version pour la contrainte anti-monotone.*

Pour identifier une solution de longueur n, chacun de ces deux étapes aura besoin jusqu'à n passes sur la base de données, ce qui peut ne pas être optimal. Le désir de continuer l'optimisation a conduit au développement de FAVST.

Favst

FAVST succède a VST [32]. Cet algorithme utilise un treillis de suffixes pour piloter le processus d'extraction. A la différence des arbres suffixes classiques [99, 102], FAVST apporte quelques modifications :

- il comporte l'information pour le comptage des candidats
- la profondeur du treillis est limitée
- la présence d'un symbole absent de la contrainte réinitialise l'insertion de la chaine dans le treillis qui va recommencer depuis la racine pour les caractères après le symbole invalide.

FAVST découpe la base de données en n segments $D_1, D_2, \ldots D_n$ et les traite de façon séquentielle. Il gère un conjonction de contraintes anti-monotones $\mathcal{A} = \wedge_{i=1}^{n}\mathcal{A}_i$ et monotones $\mathcal{M} = \wedge_{i=1}^{n}\mathcal{M}_i$. Chaque segment n'est lu qu'une seule fois. Le pseudo code de FAVST donné dans la table 2.9 est inspiré de [61]. Notons que le traitement de contraintes ni monotones ni anti-monotones (e.g., des contraintes de similarités) a également été étudié pour pouvoir se ramener à l'expression d'une conjonction acceptable par FAVST [79, 80].

	Données en entrée : $D = D_1, D_2, \ldots D_n$, $\mathcal{A} = \wedge_{i=1}^{n}\mathcal{A}_i$, $\mathcal{M} = \wedge_{i=1}^{n}\mathcal{M}_i$
nr.	Produit la liste des séquences satisfaisant $\mathcal{A}$ et $\mathcal{M}$
	T $\leftarrow$ InitTree $(D_1, \mathcal{A}_1, \mathcal{M}_1)$
1	élagage des branches $\ominus$
2	pour tout $i = 2 \ldots n$
3	{
4	CountAndUnmark(T, $D_i, \mathcal{A}_i, \mathcal{M}_i$)
5	élagage des branches $\ominus$
6	}
7	retuorne T

TAB. 2.9 – *Pseudo code de FAVST.* ***InitTree*** *construit un treillis de chaines à partir des séquences de la base de données.*

Dans le pseudo-code de FAVST (tableau 2.9), la procédure **InitTree** construit un VST correspondant au segment D_1 de la base de données par une technique d'arbre suffixe classique : chaque chaine de D_1 est insère dans l'arbre dans lequel on maintient des compteurs pour connaitre exactement la fréquence de chaque suffixe. Quant on rencontre un symbole inintéressant, on recommence l'insertion de la chaine depuis la racine à partir de la position suivante. De plus, la profondeur de l'arbre est contrôlée pour limiter l'usage de mémoire. Après la construction de l'arbre suffixe, on marque toutes les branches de l'arbre par $\oplus$ et $\ominus$ en fonction de $\mathcal{A}_1$ et $\mathcal{M}_1$

Ensuite, cette structure n'est jamais augmentée par les étapes successifs de **CountAndUnmark**. Cette procédure est très similaire à **InitTree**. Lors de l'insertion des nouvelles

chaines les compteurs des nœuds du VST sont incrémentés si le suffixe a fait partie de l'arbre construite de D_1. Après le traitement de chaque segment D_i, le VST est élagué pour enlever les branches ne contenant que de $\ominus$. Ceci augmente la vitesse de traitement des segments à venir.

Compte tenu de sa structure, pour traiter une conjonction de contraintes fréquent, infréquent FAVST ne lit la collection de chaines qu'une seule fois.

2.3.5 Autres techniques

Tableaux de suffixes

L'inconvénient majeure des treillis suffixes utilisés par VST et FAVST est que, dans le cas le plus défavorable, ils peuvent occuper un espace mémoire proportionnel au carré de la taille de la base de données, ce qui les rend inadapté aux extractions de grande échelle.

L'équipe de Kramer à l'Université Technique de Munich a proposé dans [41] une méthode plus performante que FAVST, basée sur les tableaux suffixes (suffix arrays) [68]. Les tableaux suffixes sont une représentation de l'ordre lexicographique des suffixes d'une collection de chaines.

Définition 3. *Le **tableau suffixe** d'une chaine S de longueur n est un tableau d'entiers $SA[1..n]$ prenant des valeurs dans $\{1 \ldots n\}$, tel que $S_{SA[i]}..S_n \leq S_{SA[i+1]}..S_n$, pour $1 \leq i \leq n$.*
*Le **tableau suffixe généralisé** d'une base de données de chaine $DB = \{S_1, S_2 \ldots S_n\}$ est le tableaux suffixe de la chaine $S = S_1 + \sharp_1 + S_2 + \sharp_2 + S_n\sharp_n$ obtenue par la concaténation de toutes les chaines de DB. Les symboles $\sharp_1, \sharp_2$ marquent la fin des chaines individuelles, et sont lexicoraphiquement inférieurs a tout caractère de Σ. De plus, on a $\sharp_1 < \sharp_2 < \ldots < \sharp_n$.*
*Le tableau des **préfixes communs maximaux** (LCP - longest common prefix) d'une chaine S de longueur n est un tableau d'entiers $LCP[1..n+1]$ satisfaisant $LCP[i] = \max\{k \mid S_{SA[i]}..S_{SA[i]+k} = S_{SA[i+1]}..S_{SA[i+1]+k}\}$, c.à.d. la longueur du préfixe maximale commune des suffixes de S se succédant dans l'ordre lexicographique. Par définition $LCP[1] = LCP[n+1] = -1$.*
*On appellera **maximum local** le paire d'indices (l,r) satisfaisant $LCP[l-1] < LCP[l] = LCP[l+1] = .. = LCP[r] > LCP[r+1]$.*

Le tableau 2.10 illustre ces définitions par l'exemple de la base des séquences $\mathcal{DB} = \{abadcadb\sharp_1, aba\sharp_2\}$.

Pos	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Text	a	b	a	d	c	a	d	b	$\sharp_1$	a	b	a	$\sharp_2$	
SA	9	13	12	10	1	6	3	8	11	2	5	7	4	
LCP	-1	0	0	1	2	1	2	0	1	2	0	0	1	-1

TAB. 2.10 – *Démonstration des structures SA et LCP sur la base des séquences $\mathcal{DB} = \{abadcadb\sharp_1, aba\sharp_2\}$. En effet, SA[1]=9 signifie que la première sous-chaine de S (dans l'ordre lexicographique) commence à la position neuf. De même, LCP[5]=2 indique que la 5ième et 6ième sous-chaines de S - partagent deux caractères dans leurs préfixes.*

Les tableaux suffixes peuvent être construits en temps linéaire avec l'algorithme de Manber et Myers (1993) [68] ou avec la méthode plus récente de Ko et Aluru (2004) [56]. Kasai et al. (2001) [55] donne une excellente méthode pour calculer les tableau des préfixes communes maximales (LCP).

Les tableaux suffixes sont une représentation alternative des treillis suffixe. En effet, On peut s'apercevoir que qu'un parcours en profondeur (depth first) selon l'ordre lexicographique des feuilles d'un treillis suffixe génère exactement l'ordre défini par un tableau suffixe. L'extraction des sous-chaines fréquentes s'appuie sur cette analogie.

Définition 4. *La **location** loc(S) d'une chaine S dans un treillis suffixe T est le nœud N ou se termine S.*
*La position SA[i] dans le tableaux suffixe (ou la feuille correspondante dans le treillis suffixe équivalent) **couvre** la chaine $S_j \in DB$ ssi le premier caractere de fin de chaine ($\sharp$) de $S_{SA[i]..\|S\|}$ est $\sharp_j$. Par exemple, dans le tableau 2.10 SA[3] couvre abadcadb$\sharp_1$.*

Théorème 1. *(Theoreme 3 de [41]) Dans une base de chaines DB, la fréquence d'une chaine P est égale au nombre de chaines dans DB couvertes par la location de P.*

L'algorithme de Kramer construit d'abord les tableaux SA et LCP pour la base des chaines de caractères. Ensuite, en se servant des propriétés de ses deux structures, les sous séquences fréquentes sont trouvées en manipulant les *maximalocaux*. La définition formelle est donne dans le tableau 2.11

	Données en entrée : la base des chaines DB, et la séquence minimale
nr.	Produit la liste des séquences fréquentes dans DB
1	calcule SA[1..n] et LCP[1, n+1] }
2	l ← 1 et r ← 1 et res ← ∅
3	tant que r < n faire
4	(l,r) ← nouvelle maximum local à droite de r
5	cov ← card$\{S \in DB \mid SA[l-1]..SA[r]$ couvre $S\}$
6	k ← LCP[r]
7	tant que k < cov
8	k ← max LCP[l-1], LCP[r]
9	si k<0 continue ligne 4
10	tant que LCP[r+1]=k faire r← r+1
11	si LCP[r+1] > k continue ligne 4
12	LCP[l-1] ≥ k faire l ← l-1
13	cov ← card$\{S \in DB \mid SA[l-1]..SA[r]$ couvre $S\}$
14	res = res ∪ $S_{SA[r]..SA[r]+k-1}$
15	retourne res

TAB. 2.11 – *Extraction des sous-chaines fréquentes avec des tableaux suffixes.*

La vitesse d'exécution, et le passage à l'échelle de algorithme sont meilleurs que FAVST mais la méthode nes'applique qu'aux contraintes de fréquence (fréquences maximales et minimales).

Spirit

En 1999, Garofalakis et ses coauteurs ont proposé la famille d'algorithmes Spirit pour traiter des conjonctions d'une contrainte de fréquence minimale et d'une contrainte d'expression régulière [44].

Il s'agit de trouver les séquences qui sont à la fois fréquentes et acceptées comme phrases d'un langage régulier donné. Ces langages sont spécifiés par les classiques expressions régulières (ER) construites avec les opérateurs de concaténation, d'alternance et de fermeture de Kleene (zéro à plusieurs occurrences des motifs décrits dans la sous-expression sur laquelle la fermeture est appliquée). Par exemple, la contrainte $(a \cup b) * cde*$ représente les séquences débutant par une première partie (optionnelle) ne contenant que des a ou des b suivie de cd et finissant par zéro à plusieurs lettres e. Ainsi, les séquences *abbbacd, bacdee, acdeee* ou *cde* peuvent être générées à partir de cette expression régulière.

L'extraction des séquences fréquentes satisfaisant une ER est difficile. En effet, ces contraintes ne sont généralement ni anti-monotones ni monotones : par exemple, *bacdee, acdeee* et *cde* satisfont $(a \cup b) * cde*$, mais la sous-séquence *de* de *cde* n'appartient pas au langage généré par cette ER.

La recherche des motifs séquentiels correspondant à une expression régulière donnée doit exploiter conjointement l'espace de la contrainte et l'espace des données. Les travaux de Oates [86] et [87] s'appuient également sur l'exploration de l'espace de la contrainte mais cet auteur ne cherche pas l'élagage dans les deux espaces. Il peut donc arriver, que certaines parties d'une expression régulière correspondent à beaucoup de séquences dans la base de données alors que les autres sous-expressions soient plutôt pauvres en séquences générées.

Garofalakis et ses collègues ont adopté une approche d'extraction par relaxation de la contrainte de départ et filtrage a postériori des résultats obtenus. Plusieurs niveaux de relaxation sont proposés, chacun donnant des performances qui dépendent de la sélectivité expérimentale de la contrainte (nombre de séquences qui satisfont la contrainte dans la base de données). Les expressions régulières sont représentées dans Spirit par un automate à états finis. La génération des candidats s'appuie sur un critère d'élagage qui peut être poussé à l'intérieur de l'extracteur : seulement les séquences correspondant à un fragment de chemin dans l'automate sont générées. Le choix des chemins considérés lors de la génération des candidats détermine quatre algorithmes :

- **Spirit (N)** le premier niveau de l'algorithme s'appuie sur une contrainte monotone : seuls les symboles apparaissant dans la contrainte peuvent et doivent être présents dans les candidats. On est pratiquement dans le cadre de l'algorithme Apriori, la contrainte ER est renforcée à la fin de l'extraction.
- **Spirit (L)** - les candidats sont des séquences qui correspondent à des fragments de chemins dans l'automate. Ce cadre bénéficie des possibilités d'élagage à base de fréquence.
- **Spirit (V)** - les candidats sont des fragments de chemins de l'automate qui finissent par le(s) nœuds terminal(aux). L'élagage à base de fréquence ne fonctionne pas toujours, certaines sous-séquences non terminales ne sont pas générées alors qu 'elles auraient pu conduire à des élagages.
- **Spirit (R)** - les candidats sont des séquences qui satisfont complètement l'expression régulière. Il n'y a aucun élagage à base de fréquence des sous-expressions, tous les

candidats possibles sont générés et comptés.

Quand le nombre de motifs séquentiels qu'une contrainte pourra engendrer est grand, les variantes Spirit(R) et Spirit(V) donnent des résultats très médiocres. Ces contraintes auraient besoin de l'élagage à base de fréquence, mais ces algorithmes limitent totalement ou très fortement la prise en compte de cette information (beaucoup de sous-séquences potentiellement intéressantes pour l'élagage ne sont même pas générées). Spirit(L) est basé sur l'élagage à base de fréquence. Ses résultats dépassent ceux de Spirit(R) et Spirit(V) quand les contraintes ont une sélectivité théorique élevée. Cependant, dès que la contrainte devient plus précise et que le nombre de séquences qu'elle codifie diminue, on constate un retournement de tendance. Spirit(L) est alors nettement devancé par Spirit(V) et encore plus par Spirit(R). On retrouve ici les principaux problèmes rapportés dans [25, 100].

Les algorithmes Spirit ne proposent pas de contrôles basés sur la sélectivité théorique (globale ou locale) de la contrainte, leurs performances sont donc très inégales et imprévisibles, ce qui limite leur champs d'application (voir figure 2.16).

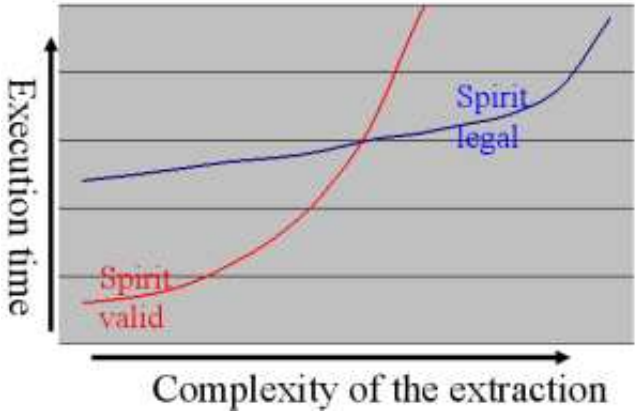


FIG. 2.16 – *Comportement des algorithmes Spirit.*

La sélectivité des contraintes est une propriété que l'on ne maîtrise pas en début d'extraction. Même si un niveau donné de Spirit pourrait donner de bons résultats, il n'y a aucun moyen qui permette de le choisir avant l'extraction. Dans le cas idéal, il faudrait avoir un algorithme stable et faiblement dépendant de la sélectivité de la contrainte, un algorithme qui s'adapte à la sélectivité individuelle des sous-expressions de la contrainte et choisisse la méthode d'élagage au niveau des sous-séquences au cours de l'extraction. Le tableau 2.12 synthétise ces considérations.

Bien qu'il y ait eu de nouveaux développements pour l'extraction des séquences correspondant aux expressions régulières (e.g., [90]), les questions posées ci-dessus restent ouvertes.

En effet, [90] montre que les techniques de projection peuvent donner de bons résultats pour les ER. Conceptuellement, cette solution est proche de SPIRIT(V) : cet algorithme construit des bases de données projetées pour les séquences qui peuvent préfixer l'ER. En fait, les automates à états finis (AEF) se prêtent mal à l'exploitation active des particularités locales de l'expression régulière correspondant à une contrainte. Cette représentation fait perdre une partie de l'information initiale. Ainsi, la structure hiérarchique de l'ex-

Contrainte ER	Spirit(N)	Spirit(L)	Spirit(V)	Spirit(R)
Sélectivité élevée	Peu d'intérêt	Bon élagage à base de fréquence	Peu d'élagage basé sur la fréquence, trop de séquence générées. Perte de l'anti-monotonie	Le comptage des séquences finales explose à la dernière étape
Sélectivité faible	Peu d'intérêt	Trop de sous séquences comptées	Bon compromis entre les deux types élagage	Peu de sous séquences comptées

TAB. 2.12 – *Comparatif des algorithmes Spirit*

pression régulière est "aplatie". Par exemple, un AEF construit à partir de l'expression $(abc \cup dct)a$ conduit à la génération des candidats $ca, ta, bca, cta, abca$ et $dcta$. Les automates n'expriment pas le fait que les fragments abc et dct sont des séquences élémentaires qui doivent apparaître ensemble. La fréquence de ct ou de bct ne permet ni de gagner des accès à la base de données ni de diminuer le nombre des candidats. Certaines chaînes partielles doivent apparaître ensemble et la connaissance de la fréquence des fragments sous-atomiques est inutile.

Un bon algorithme devrait exploiter les particularités locales de l'ER et limiter l'explosion combinatoire constatée expérimentalement avec Spirit(R). Les défauts de Spirit sont illustrés dans le tableau 2.12. En supposant que toutes les séquences sont fréquentes, les trois dernières variantes de Spirit génèreraient les candidats comme dans le tableau 2.13.

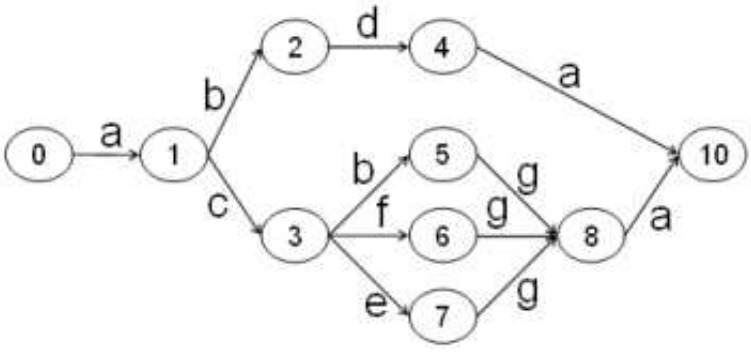


FIG. 2.17 – *Automate à états finis pour l'expression* $a(bd \cup (c(b \cup e \cup f)g))a$

Pendant la phase de comptage des candidats, les algorithmes Spirit travaillent toujours sur des candidats de même longueur, alors que dans certains cas (cf. section 4.4.2), ils pourraient aller bien plus vite. Compter des candidats de longueurs inégales dans une même passe sur les données ne pose aucun problème technique. La difficulté de l'extraction

Niveau	Spirit(L)	Spirit(V)	Spirit(R)
1	a, b , c, d e, f, g	a	-
2	ab, bd, da, ac, cb, ce, cf, bg, eg, fg, ga	a, ga	-
3	abd, bda, acb, ace, acf, cbg, ceg, cfg, bga, ega, fga	bda, bga, ega, fga	-
4	abda, acbg, aceg, acfg, cbga, cega, cfga	abda, cbga, cega, cfga	abda
5	acbga, acega, aefga	acbga, acega, aefga	acbga, acega, aefga
Candidats/Passes	39/5	14/5	4/2

TAB. 2.13 – *Génération de candidats avec les algorithmes Spirit pour l'automate représenté sur la figure 2.17.*

vient du fait que les ER ne sont pas anti-monotones, et ne présentent aucune technique évidente pour l'élagage de l'espace de recherche.

ExAMiner

Au chapitre des conjonction des contraintes, en travaillant sur les ensembles fréquents, Bonchi [23] regarde la possibilité de combiner la contrainte de fréquence avec une contrainte syntaxique qui n'est pas anti-monotone. La contrainte non anti-monotone est utilisée pour réduire la taille de la base de données à chaque génération, alors que la contrainte anti-monotone de fréquence sert à élaguer l'espace des candidats ! Notre algorithme Re-Hackle est une autre tentative pour combiner les contraintes anti-monotones avec des contraintes quelconques.

2.4 Synthèse

La recherche des motifs séquentiels est un domaine vivant, et l'état de l'art a présenté une grande diversité d'extracteurs. Une classification naturelle des propositions repose sur les différentes stratégies d'exploration des espaces de recherche. Si GSP [97] ou Spirit [44] réalisent un *parcours en largeur*, Spade/cSpade [106, 107] ou PrefixSpan [90] font un *parcours en profondeur*. Le parcours en profondeur offre la possibilité de paralléliser l'exécution et de ne travailler sur des partitions de la base sans de façon complètement indépendante. Pour des raisons historiques, la stratégie itérative qui consiste à *générer les candidats*, compter leurs fréquences, retourner les motifs fréquents est la plus répandue (Apriori, Spade, Spirit). Malheureusement, ce processus est très couteux. Les techniques de projection (freeSpan [49] et PrefixSpan [90]) offrent une alternative assez séduisante : les *manipulations successives de bases données conduisent à l'émergence de séquences fréquentes dans des bases projetées de taille réduite.*

Chacun des algorithmes a ses points forts et faibles. En fonction de la densité des données et la contrainte considérée, on utiliserait tantôt l'un, tantôt l'autre. La génération et le comptage des candidats à la Apriori a souvent un coût prohibitif, ce qui a motivé les techniques de projection et de listes d'occurrences. Ces nouvelles méthodes ont apporté de nouveaux problèmes : d'une part, le coût du calcul et du stockage des bases de données projetées est important, d'autre part le stockage dans la mémoire principale des listes d'occurrences peut poser des problèmes de passage à l'échelle. L'intégration des contraintes au sein de l'extraction afin de ne retourner que les séquences intéressant l'expert permet souvent d'améliorer la qualité de la sortie. Les contraintes peuvent apparaitre dans la phase de post-traitement, ou agir durant l'extraction, ce qui augmente l'efficacité algorithmique. Ce sont les propriétés des contraintes (anti-monotone, monotone, succincte, préfixe (anti)-monotone) qui déterminent si elles peuvent être utilisées durant l'extraction.

Les *expressions régulières* (ER) sont des contraintes syntaxiques puissantes et utiles dans de nombreuses applications. Il s'agit aussi d'un cas exemplaire pour étudier le traitement efficace de conjonctions de contraintes anti-monotones (fréquences minimales) avec une contrainte quelconque. Nous proposerons le cadre *Re-Hackle* pour l'extraction des séquences fréquentes satisfaisant une ER. L'étude de l'élagage en présence d'une ER nous a permis d'identifier la nouvelle famille des *contraintes en cascade*, qui permettent une relaxation localement anti-monotone automatique de certaines contraintes.

Chapitre 3

Techniques d'extraction par parcours d'arbres

L'étude des stratégies d'élagage lorsque la contrainte utilisée n'est pas (globalement) monotone ou anti-monotone a reçu une attention croissante au cours des dernières années. L'idée dominante consiste à relaxer la contrainte initiale par une contrainte de préférence anti-monotone plus faible pendant l'extraction et de filtrer les séquences trouvées en post-traitement. L'élagage à base de projections a également été proposé. Il est très difficile de trouver un critère de relaxation systématique autre qu'une improvisation ad-hoc. De plus, la relaxation ne doit pas se faire uniquement en fonction de la contrainte mais aussi en considérant le contenu de la base des données (BD) [6]. A notre connaissance, peu d'attention a été consacrée aux extractions pilotées conjointement par la contrainte et le contenu de la BD.

Une extraction réussie doit se concentrer sur l'exploration des plus petits sous-espaces de recherche. Au cours de nos travaux, l'extraction des séquences fréquentes a été perçue comme un processus itératif, une suite de raffinements successifs qui s'appuient sur l'information collectée jusqu'alors afin d'adapter la méthode d'extraction à l'univers de l'extraction : le contexte de l'extraction combine donc la contrainte et le contenu de la BD. Plus précisément, on considère l'itération des trois étapes :

- la détermination des (sous-)espaces de recherches qui peuvent être explorés indépendamment
- l'exploration de ces problèmes indépendants
- l'analyse de l'information collectée et détermination des nouveaux sous-espaces

Il n'est pas possible de traiter ainsi toutes les contraintes. Ce mémoire est consacré à l'étude de la famille des contraintes qui peuvent être traitées de manière efficace en itérant ces trois tâches. Ce sont nos travaux qui identifient cette famille. L'idée est d'abord née en 2003 dans [4, 5, 6] pour les contraintes exprimées par des expressions régulières (ER). Un première généralisation a été fournie dans [7] pour les expressions logiques (ou expression régulières avec des variables). Nous sommes arrivées à la conclusion que la représentation des expressions régulières par des Automates à États Finis (AEF) faisait perdre de l'information en aplatissant la structure hiérarchique présente dans la contrainte initiale. Nous avons réalisé qu'il s'agissait d'une perte importante car la structure d'une ER permet de définir des stratégies d'extraction locales. Une ER est d'abord un arbre de

dérivation de la contrainte, une succession de règles de dérivation d'une grammaire hors contexte. Les techniques de ce mémoire réutilisent l'arbre de dérivation de la contrainte (ER ou même des formes plus générales) et associent un sous-espace de recherche à chaque sous-expression de la contrainte initiale. Chaque règle de dérivation est considérée comme la combinaison de l'information, i.e., sous-séquences fréquentes, contenue dans des sous-espaces de recherche. Muni des fréquences des sous-expressions, il devient alors possible de travailler sur l'arbre de dérivation, et de re-séquencer l'exploration des sous-espaces restant afin de minimiser l'effort d'extraction. La structure utilisée est celle d'un arbre de syntaxe abstrait (introduit dans la section suivante) exploré de façon ascendante. Chaque partie de l'arbre de syntaxe abstrait codifie une partie de la contrainte et sert à piloter l'extraction. La méthode proposée s'inscrit dans le cadre de l'extraction par niveaux formalisée par Mannila et al. en 1995 [69]. A chaque génération (niveau), certains nœuds de l'arbre sont considérés, génèrent des candidats, et contribuent - après comptage des candidats - à la transformation de l'arbre d'extraction. Les transformations d'arbres incluent évidemment l'élagage - les branches qui ne pourront plus donner des séquences fréquentes sont retirées - mais aussi des unions ou scissions des espaces de recherche, s'il apparaît que deux sous-espaces peuvent être traités de façon plus efficace ensemble (e.g., au lieu de compter séparément la fréquence de a, b et ensuite ab, il est quelques fois possible, et plus rentable, de traiter directement ab) ou bien un espace de recherche trop important sera découpé pour réaliser plusieurs recherches partielles à des coûts d'extractions moindres. Le cadre algorithmique proposé - Mining Cascaded Constraints (MCC) - sait faire ces choix. Il est également possible d'introduire une nouvelle génération s'il est plus rentable de renforcer l'élagage à base de fréquence avant de reconsidérer la contrainte.

Passons à la définition de la famille des contraintes en cascade et à la description d'un principe algorithmique d'extraction. Il sera ensuite montré que les expression régulières sont un cas particulier des contraintes en cascade et que la concaténation, l'alternance et la fermeture de Kleene correspondent aux opérations d'unification, de réduction et de scission des espaces de recherche.

3.1 Définitions

Arbre abstrait de syntaxe - abstract syntax tree (AST). L'AST est la structure pour représenter la contrainte et pilote la stratégie d'extraction. Initialement, l'AST décrit le langage auquel toute séquence candidate doit appartenir. C'est l'espace de recherche maximal. Au cours de l'extraction, toute nouvelle information est intégrée à l'AST ce qui sert à générer les candidats. L'AST remis à jour à chaque génération représentera plus que l'espace des candidats, il est l'espace des candidats possibles, sachant les fréquences des candidats déjà comptés ! La représentation d'une contrainte par AST n'est pas unique. Chaque arbre détermine une stratégie d'extraction, et évolue au cours de l'extraction.

Dans nos travaux, un AST représente une contrainte si et seulement si
- les feuilles contiennent les sous-espaces de recherche (e.g., des séquences)
- les nœuds internes contiennent un triplet : *transformation élémentaire*, *fonction génératrice*, et le *contexte d'extraction*.

Le **contexte d'extraction** contient l'information déduite des précédentes passes sur la BD : le nombre de candidats fréquents (si connu), le nombre de candidats potentiels.

Le contexte inclut également la variable *Exploration* tri-valuée :

- *enAttente* - ce nœud n'a pas encore été considéré pour la génération des candidats. Ses sous-expressions ne sont probablement pas encore connues.
- *enCours* - ce nœud est en train de générer des candidats. Ceci peut durer plusieurs génération pour une fermeture de Kleene, ou une seule génération pour une concaténation.
- *finie* - l'exploration du nœud est terminée.

et la variable *Etat* tri-valuée :

- *indéterminé* - ce nœud n'a pas encore été considéré pour la génération des candidats. Ses sous-expressions ne sont probablement pas encore explorées.
- *satisfait* - l'exploration du sous-espace de ce nœud a trouvé quelques séquences fréquentes qui participeront à la construction de séquences candidates dans les prochaines générations.
- *violé* - le sous-espace de recherche associé à ce nœud est vide, le nœud ne pourra plus participer à la construction de séquences candidates et la branche de l'arbre le contenant doit être élaguée.

Les **fonctions génératrices** fournissent des candidats en considérant le sous-espace de recherche associé à un nœud de l'AST. L'évaluation complète d'une fonction génératrice peut durer plusieurs générations, et nécessiter plus d'une passe sur la BD. Le **rang** d'une fonction génératrice est k si et seulement si k est la première génération au cours duquel la fonction pourra générer des séquences candidates à compter durant la k^{ieme} passe sur la BD (cf. figures 3.1 et 3.2). Le domaine d'une fonction génératrice est l'union ou le produit cartésien des sous-espaces du nœud de l'arbre le contenant, et son application prend des valeurs dans les séquences de symboles de l'alphabet Σ.

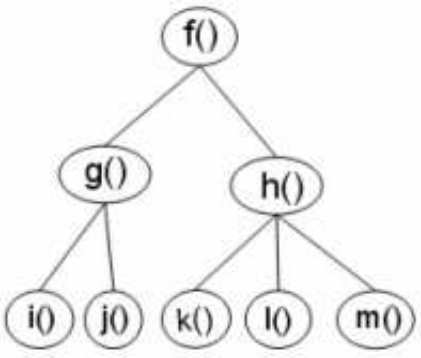


FIG. 3.1 – *Dans cet exemple générique, chaque nœud de l'AST contient une fonction génératrice. Les fonctions des niveau supérieurs $g()$ et $h()$ et puis $f()$ peuvent générer leurs candidats une fois que les séquences fréquentes sont connues aux niveaux inférieurs. Ainsi, seulement $i()$, $j()$, $k()$, $l()$ et $m()$ peuvent être comptées dans la première génération, (leur rang est au moins 1). $g()$ et $h()$ doivent attendre que tous leurs fils commencent à leur envoyer des séquences fréquentes. ($rang(g) \geq 1 + \max(rang(i), rang(j))$, $rang(h) \geq 1 + \max(rang(k), rang(l), rang(m))$), et enfin $f()$ a besoin des données du $g()$ et $f()$, donc son rang est au minimum* 3.

Fonction génératrice en cascade. Les fonctions de rang 1 sont 1-cascadées. Une fonction est k-cascadée si son domaine est donné par l'union ou le produit cartésien de fonctions génératrices $k-1$-cascadées. L'inclusion d'une fonction f dans g signifie que f se trouve dans le sous-arbre de g. Cette situation est notée $g \subset f$.

AST en cascade. Un AST représentant une contrainte est en cascade si la fonction génératrices dans sa racine est en cascade pour un k donné (e.g., cas de la figure 3.1). Une contrainte est en cascade si et seulement si elle peut être représentée par un AST en cascade.

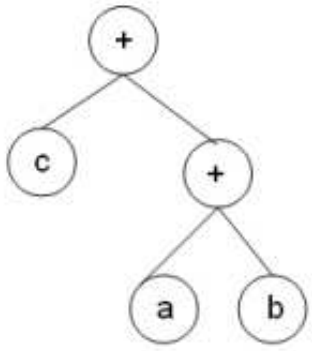


FIG. 3.2 – *Dans l'expression régulière $c \oplus (a \oplus b)$, la concaténation de c à la concaténation de a et b a un rang trop élevé. Les séquences a, b, c sont des fonctions génératrices de rang 1. Donc le rang $a \oplus b$ est 2. Le rang de la racine est 3, et demande donc trois passes sur la BD. On comprend pourquoi il est plus avantageux de transformer l'arbre pour traiter abc en une seule passe.*

Une fonction génératrice f est *anti-monotone en cascade* si et seulement si $\forall f \subset g \wedge Image(f) = \emptyset \Rightarrow Image(g) = \emptyset$. Cette définition de l'anti-monotonie est plus faible que la variante répandue dans la littérature. Les fonctions génératrices d'un rang donné peuvent être anti-monotones sans l'anti-monotonie de la contrainte globale. Remarquons, que si les contraintes sont représentées par un AST et évaluées de façon ascendante, un élagage efficace a seulement besoin de l'anti-monotonie des fonctions génératrices. On voit que les ER sont anti-monotones en cascade, et permettent donc un élagage performant.

Transformation. La transformation de l'arbre (ou sous-arbre) T résulte en un ou plusieurs nouveaux arbres. Toutes les transformations sont possibles à condition que l'union des séquences fréquentes renvoyées par le/les nouveau(x) arbre(s) coïncide avec les séquences fréquentes représentées par T. Prenons la fonction f de la racine de T. La transformation de T va permettre de sub-diviser ou d'unifier les sous-espaces de T :

- union ou produit cartésien des sous-espaces
- union ou produit cartésien de f appliqué aux sous-espaces
- f appliqué à l'union ou produit cartésien des sous-espaces
- f appliqué à l'union ou produit cartésien de f appliqué aux sous-espaces

La **duplication de l'arbre** T est une transformation d'arbre très efficace. Soit T un arbre en cours d'extraction, et N un nœud qui pourra générer des candidats durant

plusieurs passes sur la BD. Par exemple, N pourrait contenir une fonction génératrice qui produirait successivement la liste de toutes les séquences contenant des $aa\ldots$. Dans de nombreux cas, le parent P de N pourrait utiliser les séquences fréquentes déjà extraites par N avant que l'exploitation complète de N ne soit finie. Si P est, par exemple, une concaténation de $b + N.items$, les candidats $ba, baa, etc.$ pourraient être comptés avant même que la liste complète des $aa..$ soit connue. Mieux encore, si baa n'était pas fréquent, on devrait arrêter l'extraction des $aa..$ afin de terminer l'exploration (inutile) d'un sous-espace de recherche. Cela se fait un créant une copie T' de l'arbre initiale dans laquelle le nœud N est remplacé par N' qui est exploré, et contient les séquences fréquentes de $N.items$ déjà identifiées. Tous les parents $P'_1, P'_2, ..root'$ directs et indirects de N' garderont un lien avec leur homologue dans T. L'exploration de T' se poursuivra en parallèle avec celle de T, en introduisant une nouvelle règle d'élagage : Si un parent P_i du nœud générant la duplication de l'arbre est violé dans T', le même nœud sera violé dans T également. Ceci arrêtera l'exploration inutile des prochaines générations des séquences de N qui seront de toutes les façons élaguées dans l'un des parents P_i.

Par exemple dans le cas des fermetures Kleene (définies par l'équation 2.1), la duplication de l'arbre est réalisée par la procédure Dupliquer présentée dans la table 3.1

nr.	ASTListe Dupliquer(AST, k)
1	pour tout $N \in AST.\Psi_k$
2	si $(N.type = *) \wedge (N.cloned < k)$
3	retourne Dupliquer(Trans(AST, N, gauche)) $\cup$ Dupliquer(Trans(AST, N, droite))

TAB. 3.1 – *Le cadre CCM est un schéma de génération d'algorithmes qui doit être instancié avec une contrainte donnée. L'algorithme Re-Hackle du chapitre suivant est une illustration comment CCM peut s'appliquer aux expressions régulières*

Faute d'algèbre de transformations d'arbres, la procédure Trans(AST, Nœud, gauche/droite) appelée depuis la duplication d'arbres est définie graphiquement par la figure 3.3

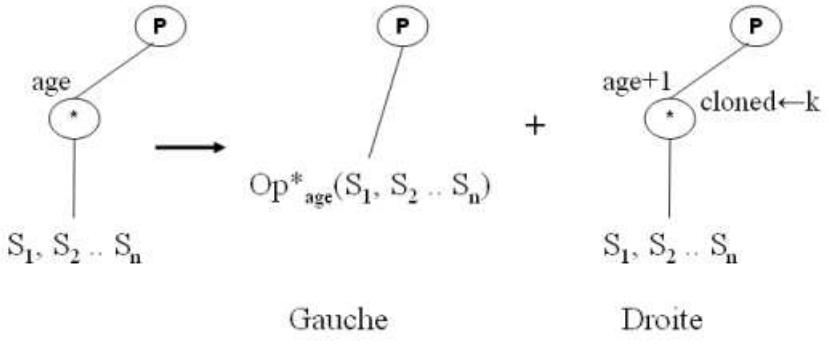


FIG. 3.3 – *La duplication d'arbre pour les fermeture Kleene crée deux nouveaux arbres.*

Sur la figure 3.3, dans l'arbre à gauche la fermeture de Kleene est remplacée par les éléments fréquents de la passe précédente, à droite on continue l'exploration de la fermeture dans une nouvelle génération (age+1). En début de l'extraction la variable *age*

est initialisée à -1, et elle sera augmentée à chaque duplication pour que les fonctions génératrices de la fermeture de Kleene (cf. equation (4.1)) puissent fonctionner.

Un détail technique, le paramètre *cloned* est nécessaire dans la duplication de la figure 3.3 pour éviter de re-dupliquer l'arbre pour le même nœud dans un contexte de plusieurs fermetures de Kleene pour le même arbre (car la procédure Dupliquer est récursive).

L'opérateur $Op*_{age}$appliqué sur un ensemble de séquences est défini comme

$$Op *_{age} (S_1, S_2 \ldots, S_n) = \begin{cases} \{\epsilon, S_1, S_2 \ldots, S_n\} \text{ si } age = -1 \\ Op \oplus_{age} (S_1, S_2 \ldots, S_n) \text{ si } age >= 0 \end{cases} \quad (3.1)$$

L'application de la duplication dans le sens de la table 3.1 et figure 3.3 sur un arbre avec deux fermetures Kleene connectées à la même concaténation génèrera les trois arbres de la figure 3.4.

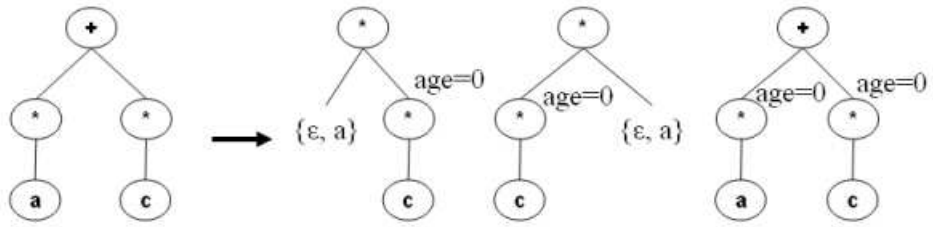


FIG. 3.4 – *Duplication d'un arbre avec deux fermetures de Kleene.*

3.1.1 Cascaded Constraint Miner

Introduisons l'extraction des contraintes en cascade au niveau conceptuel. La section suivante donne un exemple concret une contrainte en cascade, et le chapitre 4 instanciera CCM poue les expressions régulières. L'algorithme d'extraction (**Cascaded Constraint Miner, CCM**) est présenté dans la table 3.2.

nr.	L'algorithme Cascaded Constraint Miner
1	k = 1 ; $AST \leftarrow$ constraint
2	pour tout $N \in AST$ N.exploration = enCours
3	$AstListe = \{AST\}$
4	tant que $AstListe \neq \emptyset$
5	{
6	pour tout $AST \in AstList$, tel que AST.root.exploration $\neq$ finie
7	{
8	genFuncs $\leftarrow$ toutes les fonctions k-cascadées
9	pour tout $f \in genFuncs$: candidats $\leftarrow$ f()
10	compte(candidats, DB)
11	élaguer AST
12	$AstListe = AstListe \cap AST \cup$ transformer(AST)
13	}
14	k = k+1
15	}
16	retourne AST.root.items

TAB. 3.2 – *Le cadre CCM est un schéma de génération d'algorithmes qui doit être instancié avec une contrainte donnée. L'algorithme Re-Hackle du chapitre suivant est une illustration comment CCM peut s'appliquer aux expressions régulières*

D'abord, CCM construit l'arbre AST représentant la contrainte initiale (lignes 01-02), et le met dans la liste des arbres à extraire (ligne 03). Par la suite, CCM explore l'espace de recherche de façon ascendante dans un cadre par niveaux (générations) (lignes 6 - 13). S'il est possible d'accélérer l'exploration de l'espace de recherche en réorganisant l'arbre, ou s'il faut traiter des fonctions qui conduisent à la duplication de l'arbre, l'AST en cours est retiré de la liste des AST et remplacé par sa transformation (ligne 12). La transformation de l'AST peut donner lieu à plusieurs arbres qui remplaceront l'arbre initial. D'une génération à l'autre, l'étape de transformation utilise de plus en plus d'information sur la fréquence des candidats et permet d'avoir un arbre adapté à la BD.

A chaque génération, l'algorithme inspecte l'arbre, et élague les branches qui ne pourront plus contribuer à la génération des séquences satisfaisant la contrainte initiale. L'algorithme maintient la liste des nœuds qui doivent être examinés à chaque génération - appelé *phrase d'extraction* Ψ - , et produit la liste des fonctions génératrices $genFuncs$. L'évaluation des fonctions qui se trouvent dans $genFuncs$ produit les séquences candidates (ligne 9). Après le comptage des candidats (ligne 10) qui requiert une passe sur la BD, l'arbre est élagué (ligne 11) pour enlever les branches improductives. Quant l'exploration de l'AST est terminée, les séquences fréquentes se trouveront dans la racine de l'arbre.

Le cadre CCM est puissant, parce qu'il peut combiner différents contraintes (en cas-

cade !) dans une seule extraction. Par exemple, l'utilisateur pourrait chercher des séquences qui débutent par une expression régulière, et se terminent par un suffixe quelconque dont la somme ne dépasse pas une limite donnée. A notre connaissance, ce type de combinaison de contraintes n'a pas encore été proposé dans la littérature.

3.2 Exemple d'exécution

Cette exemple illustre l'extraction de la contrainte $a^n bc^n$ sur la base de séquences de la figure 3.5 (gauche). Noter qu'il s'agit d'un langage hors contexte (cf. par exemple Wilhelm et Mauer[104] pour la classification des grammaires).

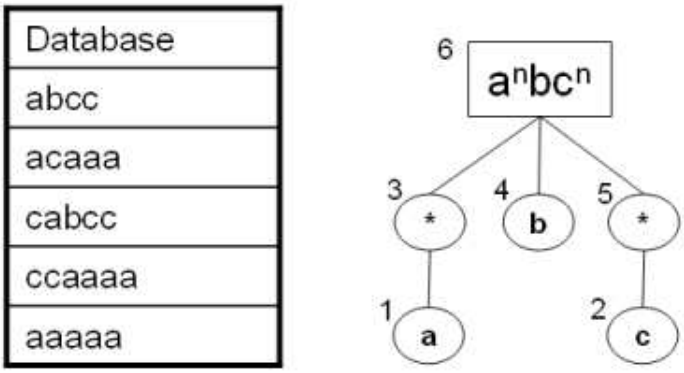


FIG. 3.5 – *Base de séquences et AST pour la contrainte* $a^n bc^n (n = 2)$.

Première étape La première étape s'occupe de l'extraction des 1-séquences. La phrase d'extraction contient les nœuds $\Psi_1 = \{1, 4, 5\}$. Après une passe de la base de séquences, on trouve que les sous-séquences a, b, c sont fréquentes. Ensuite, les fermetures de Kleene produisent une duplication de l'arbre (ligne 12) : lors de l'analyse de la génération k, l'un des nouveaux arbres contient la liste des séquences fréquentes déjà trouvés avant le comptage de la $k$ième génération, l'autre reprend la fermeture Kleene, avec les séquences fréquentes déjà identifiées jusqu'ici (y compris la génération k). Le résultat final est illustré à gauche de la figure 3.6. Les trois nouveaux arbres et l'arbre initial (stockés dans $AstListe$) sont passés à la génération suivante.

Seconde étape La deuxième génération doit traiter les quatre arbres de la génération précédente. Commençons d'abord avec le traitement des petits arbres de la figure 3.6, dénotés A, B, et C de gauche à droite. La phrase d'extraction de 3.6 :A contient seulement la racine de l'arbre $\Psi_{2A} = \{6\}$. La séquence b satisfait $a^n bc^n (n = 0)$, et elle sera donc produite parmi les séquences satisfaisant la contrainte globale. De gauche à droite, on a $\Psi_{2B} = \{3\}$ et $\Psi_{2C} = \{5\}$. Ces arbres ne pourront jamais générer de séquences satisfaisant $a^n bc^n$, autre que b puisque les sous-arbres à gauche ou à droite sont vides (ϵ).

La phrase d'extraction de l'arbre principal devient $\Psi_2 = \{3, 5\}$, ce qui permet d'identifier les sous-séquences fréquentes aa, cc, satisfaisant les contraintes locales $a*$ et $c*$. La

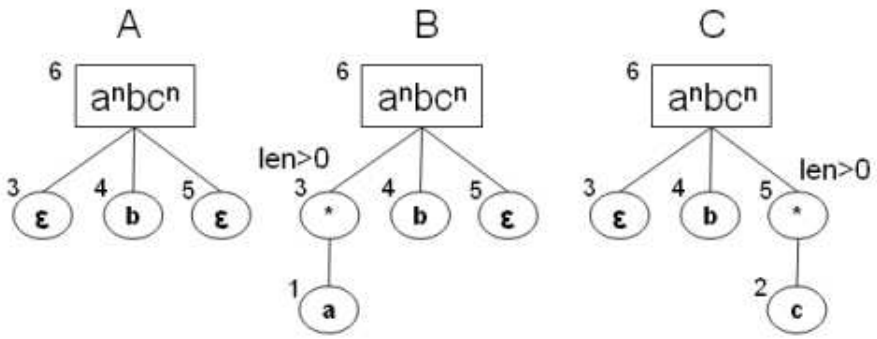


FIG. 3.6 – *Exemple d'exécution de CCM - étape 1*

transformation de l'arbre donnera les nouveaux arbres de la figure 3.7 également dénotés A, B et C (de gauche à droite). On remarquera que les branches correspondant à des fermetures de Kleene seront contraintes à ne renvoyer que les sous-séquence de longueur supérieure à 1 (ϵ étant traite auparavant).

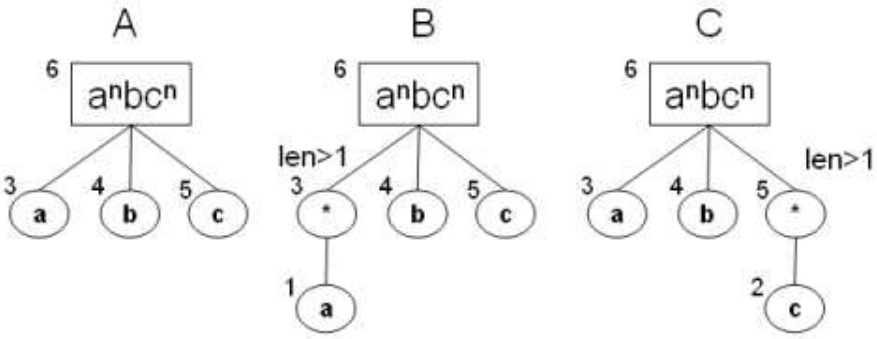


FIG. 3.7 – *Exemple d'exécution de CCM - étape 2*

Troisième étape L'exploration de l'arbre 3.7 :A ($\Psi_{3A} = \{6\}$) permet d'identifier que *abc* satisfait la contrainte globale (avec $n = 1$). Les deux arbres 3.7 :B et 3.7 :C ne pourront pas générer autre chose que *abc* et disparaitront de la liste des AST à la fin de la génération 3. La phrase d'extraction de l'arbre initial reste $\Psi_3 = \{3, 5\}$ pour trouver les sous-séquences aaa, ccc.

Quatrième étape L'arbre dérivé 3.8 :A fournit le candidat *aabcc* qui n'est pas fréquent. Le nœud 6 est donc violé. Les deux autres arbres 3.8 :B, 3.8 :C génèreront le même candidat. Tous les arbres dérivés étant violés, l'algorithme enlève l'arbre principal de *AstListe*. Ceci termine l'extraction. Les séquence fréquentes satisfaisant $a^n bc^n$ sont donc b et abc.

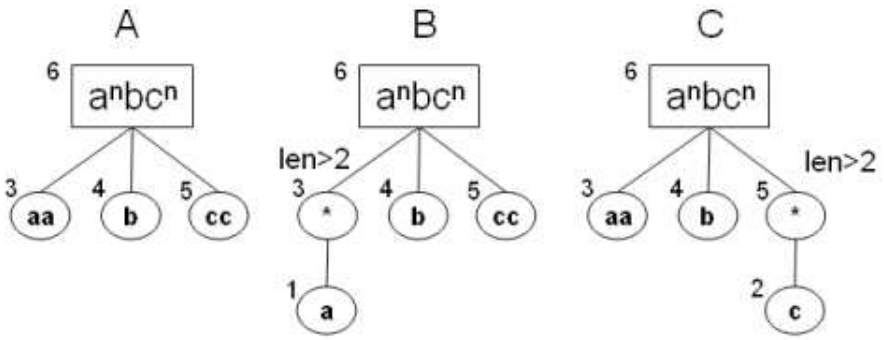


FIG. 3.8 – *Exemple d'exécution de CCM - étape 3*

Remarque Un candidat comme *aaaa* n'est même pas généré : l'information recueillie sur la contrainte jusqu'à la génération 4 permet de dire que même si *aaaa* est fréquente, elle ne pourra pas contribuer à la contrainte locale. C'est un exemple typique d'élagage rendu possible par notre méthode qui combine plusieurs contraintes en cascade poussées dans l'extracteur.

3.3 Conclusion

Historiquement, le cadre Re-Hackle a précédé l'introduction des contraintes en cascade. Les idées de ce chapitre confirment l'intuition que la représentation arborescente des conjonction des contraintes s'applique à une famille de contraintes plus générale que les expressions régulières. Si une contrainte peut être découpée en une série de contraintes élémentaires imbriquées qui satisfont certaines propriétés (cascade), le cadre CCM permet d'instancier un algorithme pour traiter l'élaguage de façon très efficace. Le chapitre suivant présente justement l'algorithme Re-Hackle qui est une adaptation de la méthode des contraintes en cascade au cas particuliers des contraintes exprimées par des expressions régulières.

Chapitre 4

Le cadre Re-Hackle

Fort des idées avancées lors de la présentation de Spirit, et exploitant le cadre des contraintes en cascade (chapitre 3), nous proposons l'algorithme Re-Hackle (Regular Expression Highly Adaptive Constrained Local Extractor) [5, 6] qui permet le pilotage de la méthode d'extraction en fonction des particularités locales (i.e., les sous-expressions) de la contrainte. L'exploration conjointe de l'espace des séquences et de la base de données permet d'exploiter activement les deux types d'élagages : l'élagage à base de fréquence et celui qui est lié à la contrainte exprimée par l'expression régulière (ER).

Lors du développement de Re-Hackle, nous avons étudié plusieurs représentations des ER : notre structure de données est un arbre de syntaxe abstrait (AST) qui peut contenir plus d'informations sur les propriétés locales de la contrainte que les automates à états finis. Ces informations contextuelles sont utilisées pour analyser chaque sous-expression de l'ER initiale, et l'algorithme Re-Hackle choisit une stratégie d'extraction locale pour favoriser tantôt l'élagage basé sur l'ER tantôt celui basé sur la fréquence.

Cette démarche généralise le cadre Apriori [71], dans la mesure où les candidats au niveau k ne sont pas forcément de longueur k. Ceci permet d'assembler plus rapidement les séquences finales qu'une classique approche par niveaux. Ceci permet aussi de limiter le nombre de passes sur la base de séquences.

Ces propriétés intéressantes sont possible grâce à l'assemblage ascendant des séquences candidates. Chaque nœud de l'AST (appelé ici arbre Hackle), codifie une partie de l'expression régulière. Les nœuds contiennent des informations contextuelles provenant de l'exploration de la sous-expression qu'ils représentent. Ces informations sont ensuite utilisées pour choisir la meilleure stratégie d'extraction disponible, et réduire le temps d'exécution de l'algorithme. De plus, comme l'exemple d'exécution pas-à-pas de l'algorithme CCM (Table 3.2) l'a montré, l'arbre Hackle peut-être transformé dans certaines occasions (voir section 4.3). Les transformations possibles incluent l'élagage des branches infertiles ou le regroupement des espaces de recherche trop petits qui peuvent donc être explorés ensembles. L'algorithme Re-Hackle propose même l'insertion des nouveaux niveaux dans l'arbre lorsque le nombre de candidats explose et qu'il parait utile de renforcer l'élagage basé sur la fréquence.

La gestion des expressions régulières demande de compléter les définitions de la section 2.1 avec de nouveaux concepts.

Formellement, l'*Expression Régulière (ER)* est un élément du langage défini par la grammaire donnée dans la table 4.1.

Expr	$\rightarrow$	Term
	$\rightarrow$	Term $\cup$ Concat
Term	$\rightarrow$	Séquence ($S \in \Sigma^n$)
	$\rightarrow$	Kleene
	$\rightarrow$	(Expr)
Concat	$\rightarrow$	Concat $\oplus_k$ Term
	$\rightarrow$	Term
Kleene	$\rightarrow$	Expr *

TAB. 4.1 – *Grammaire des expressions régulières.*

Une **sous-expression** d'une expression régulière est une sous-chaîne qui vérifie également la grammaire de la table 4.1. Une **sous-expression maximale** est une sous-expression qui n'est contenue dans aucune sous-expression excepté la contrainte de départ. Une expression peut avoir plusieurs sous-expressions maximales. Par exemple, $(a \cup b)$ et $(c \cup b)*$ sont les deux sous-expressions maximales de l'expression $(a \cup b) \oplus_k (c \cup b)*$.

Une **séquence atomique** est une sous-expression de la contrainte initiale qui ne contient aucun opérateur (union, concaténation ou fermeture de Kleene). L'ER est définie sur les séquences atomiques. Remarquons qu'ils existent des concaténations implicites entre les symboles d'une séquence atomique, mais nous ne les prenons pas comme des opérateurs. Par exemple, $abd \oplus ef$ est composé de deux séquences atomiques abd et ef, la concaténation implicite dans abd - $(a \oplus b \oplus d)$ - et ef - $e \oplus f$ - n'est pas mentionnée dans la grammaire qui fait apparaitre les séquences atomiques au même titre que les symboles de l'alphabet initial.

L'**opérateur prioritaire** réunit les sous-expressions maximales pour former la contrainte globale. C'est l'opérateur qui a la moindre priorité. Par exemple, $\oplus$ est l'opérateur prioritaire de $(a \cup b) \oplus (a \oplus (d \cup c))$, car il relie les sous expressions maximales $(a \cup b)$ et $(a \oplus (d \cup c))$.

Une séquence est **légale** par rapport à la contrainte de départ s'il existe une sous-expression de l'expression de départ qui peut la générer. Cette définition de légalité est plus contraignante que celle de [44] pour Spirit(L). Par exemple, Spirit considèrerait les séquences ab ou de comme légales par rapport à l'expression $((a \cup c) \oplus bd \oplus (e \cup d))$. Selon nous, bd et $abde$ sont légales, mais ab ne l'est pas, car elle ne fait pas partie de l'arbre de dérivation de cette contrainte.

L'expression régulière est représentée par un arbre dans lequel chaque nœud interne correspond à un opérateur et chaque feuille terminale à une séquence atomique. Les longueurs des séquences atomiques associées aux feuilles sont inégales. L'expression régulière est composée de ces séquences atomiques assemblées à l'aide des trois opérateurs précédents. Les règles d'assemblages sont exprimées par l'arbre. La conséquence de l'arité variable fait que l'expression $a \cup b \cup c \oplus b \oplus a*$ donnera l'arbre A de la figure 4.1.

L'extraction se propose d'extraire de la base de données toutes les séquences fréquentes légales par rapport à l'arbre. L'avantage de la méthode est qu'elle ne génère que les candidats correspondant à des sous-expressions codifiées par l'arbre, ainsi l'exploration d'un grand espace de recherche est évitée. Nous suivons deux objectifs. L'objectif principal (algorithmique) est d'élaguer l'arbre Hackle qui pilote la génération des candidats en tenant

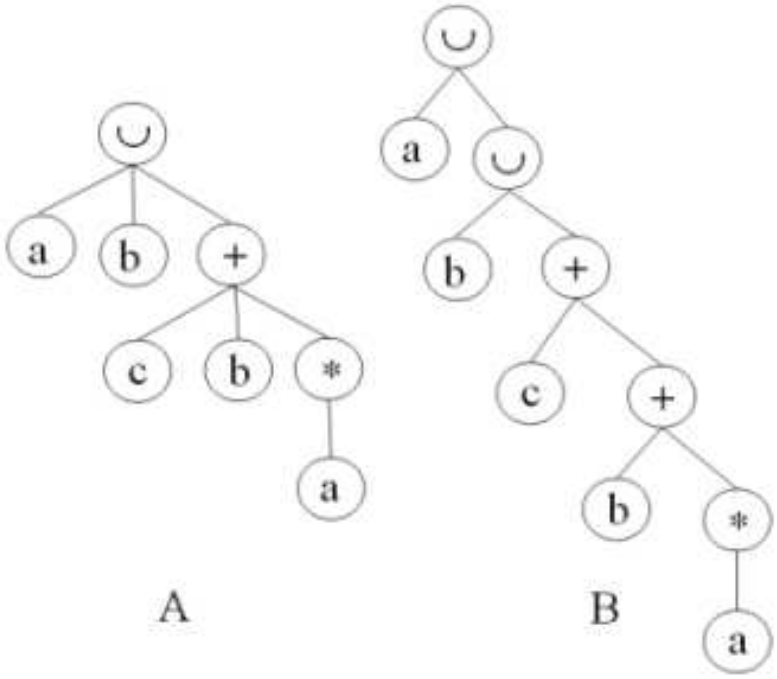


FIG. 4.1 – *L'arbre généré à partir l'expression* $a \cup b \cup c \oplus b \oplus a*$ *par Re-Hackle (A) et par l'analyseur d'un compilateur naïf (B) sont différents. L'arité variable permet de limiter la hauteur de l'arbre et d'accélérer l'extraction.*

compte à la fois de la contrainte de fréquence et de l'ER. Le second objectif est de nature méthodologique : proposer et utiliser une caractérisation de la contrainte ER par le biais des cardinalités théoriques et expérimentales.

L'arbre codifiant l'expression est traité de façon itérative, chaque étape contenant une génération de candidats. Après le comptage de leurs fréquences, les candidats fréquents iront alimenter la génération des candidats suivants qui seront assemblés par les nœuds du niveau supérieur de l'arbre. On reconnait le cadre des contraintes en cascade du chapitre précédent. La méthode reste dans le cadre des algorithmes par niveaux. Le nombre de passes de la base de séquences est donné par la hauteur de l'arbre (à laquelle, pour être plus précis, on ajoute le nombre de fois que les fermetures de Kleene sont évaluées). Ce nombre est en général inférieur ou égal à la valeur obtenue par un algorithme de type Apriori (cf. discussion du paragraphe 4.3.2).

En fonction des résultats de l'étape précédente, l'arbre est transformé à chaque étape. Le but est de repérer et couper les branches qui ne peuvent plus conduire à une séquence fréquente. Cette information est propagée au niveau supérieur à des fins d'élagage. Concrètement, si une sous-expression E_i de $E = E_1 \oplus E_2 \ldots \oplus E_n$ ne contient aucune séquence fréquente, on dit qu'il est violé, son nœud parent E ne pourra plus contenir aucune séquence fréquente, et l'évaluation des autres sous expressions $E_1 \ldots E_n$ deviendra inutile.

4.1 Exemple d'exécution

Avant d'introduire formellement l'algorithme, considérons un exemple d'exécution. Prenons l'expression régulière $a(g \cup c)g(a \cup t(a \cup (g(a \cup c)) \cup c))$, un support minimal de 2, et la base de séquences donnée sur la figure 4.2 (à droite). L'arbre correspondant à l'expression de départ se trouve sur la même figure à gauche. Afin de faciliter les explications, les nœuds des arbres sont numérotés.

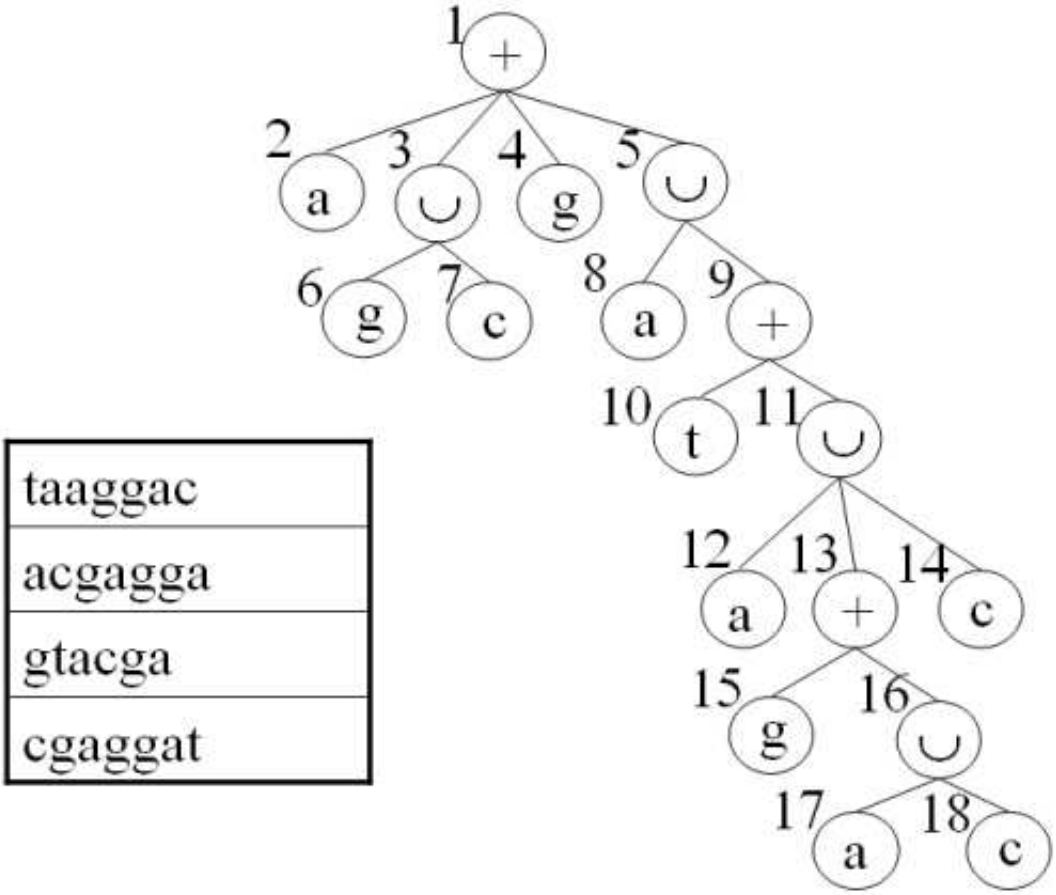


FIG. 4.2 – *Un exemple d'exécution de Re-Hackle.*

L'extraction débute par la construction de la *phrase d'extraction* Ψ_1 initiale, qui contient toutes les feuilles de l'arbre Hackle. En opérant de façon itérative, une fois que tous les nœuds dans Ψ_1 ont généré des candidats, l'algorithme construit une nouvelle phrase d'extraction, en remplaçant chaque nœud par son parent, selon les mécanismes décrits dans la section 4.2.

Pour une phrase d'extraction donnée, les candidats sont générés en visitant un par un les nœuds dans la phrase. Une *fonction génératrice* est appliquée au nœud, qui fournit la liste des candidats.

Initialisons la phrase d'extraction par les feuilles de l'AST : $\Psi_1 = \{$ *2, 6, 7, 4, 8,*

Génération	Phrase d'extraction	Candidats et fréquences
1	$\Psi_1 = \{$ *2, 6, 7, 4, 8, 10, 12, 15, 17, 18, 14* $\}$	a(4), c(4), g(4), t(1)
2	$\Psi_2 = \{3, 5\}$	aucun accès BD
3	$\Psi_3 = \{1\}$	agga(3), acga(2)

TAB. 4.2 – *Exemple d'exécution de Re-Hackle. Le résultat de l'extraction est :* $agga(3), acga(2)$

10, 12, 15, 17, 18, 14 }. Cela donne les candidats *a, g, c,* et *t*. Le tableau 4.2 illustre le déroulement de l'exécution.

On voit le fonctionnement de l'élagage après la première génération au niveau du nœud 10 qui viole ensuite la contrainte représentée par le nœud 9, i.e., son parent. Après cet élagage, l'arbre est représenté sur la figure 4.3.

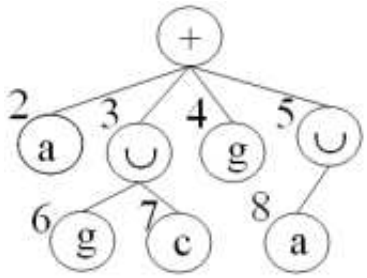


FIG. 4.3 – *Exemple d'exécution de Re-Hackle (élagage car t n'est pas fréquent).*

L'exploration des deux unions pendant la seconde génération n'a besoin d'aucun accès à la BD. La troisième génération se charge de l'exploration de la concaténation dans la racine, et trouve les séquences fréquentes *agga(3), acga(2).*

Remarque. On constate que les candidats de la génération 3 ne sont pas forcément de longueur 3, et que Re-Hackle a nécessité seulement 2 passes sur la BD pour identifier toutes les séquences fréquentes (de longueur 1 à 4). La construction de l'automate à état finis correspondant à l'expression nous montre que dans le cas où t n'est pas fréquent, tous les algorithmes Spirit génèrerait plus de candidats que l'algorithme Re-Hackle. Cet exemple simple illustre que, quand une stratégie d'élagage globale est insuffisante, la meilleure exploitation de la structure de l'expression par Re-Hackle est intéressante.

4.2 Définition formelle de Re-Hackle

L'élément central de Re-Hackle est un arbre de syntaxe abstrait, appelé **arbre Hackle**, qui représente la contrainte, et stocke aussi l'information utile au pilotage de l'extraction. L'arbre Hackle est construit comme indiqué par la table 4.5. Chaque nœud de l'arbre possède les propriétés énumérées dans le tableau 4.3.

Attribut	Signification
Type	Type du nœud : f feuille ; $\oplus$ concaténation ; $\cup$ union et * fermeture de Kleene.
Parent	Le nœud parent, NULL pour la racine
Fils	Liste des nœuds fils. NULL pour les feuilles.
Items	L'ensemble des motifs fréquents du nœud.
Etat	Etat du nœud, valeurs prises : *Satisfait, Violé, Indéterminé.*
Methode	Permet de choisir localement la méthode d'extraction spécifique à ce nœud.
Exploration	exploration du nœud est terminée ou enAttente ? (utilisée pour Kleene)
K	Paramètre pour la concaténation k-téléscopée (si N.Type=$\oplus$)
Age	Nombre de fois que le nœud a été itéré. (si N.Type=*)
Seq	La séquence atomique associée à une feuille (si N.Type=f)

TAB. 4.3 – *Définition des attributs des nœuds de l'arbre Hackle. Pour les n œds de type* $\oplus, \cup$ *la variable* ***Etat*** *devient Violé s'ils ne génèrent aucun motif fréquent après le comptage des candidats ; Satisfait signifie que l'exploration du nœud a produit quelques séquences fréquentes ; Indéterminé indique que le nœud n'a pas encore été exploré. Les fermeture Kleene sont toujours satisfaites à cause de la chaine vide* ϵ *qui est toujours fréquent.*

La **phrase d'extraction** Ψ est l'ensemble des nœuds de l'arbre qui doivent être examinés pendant chaque passe sur la base de séquences. Initialement, Ψ contient toutes les feuilles de l'arbre, énumérées de gauche à droite. Cette liste est actualisée à chaque génération en remplaçant les nœuds des niveaux inférieurs explorés par leurs parents. La phrase d'extraction ne contient jamais des nœuds en relation (directement ou indirectement) de parent-fils.

Les **fonctions extractrices** C appliquées sur un nœud N de l'arbre Hackle donnent l'ensemble des séquences candidates générées par ce nœud. Les fonctions extractrices accèdent à tous les attributs de N. Supposons $M_i \in N.fils$. Ces fonctions sont définies par l'equation (4.1).

$$C(N) = \begin{cases} \emptyset, \text{ si } \exists M_i \text{ , tel que N.etat != Satisfait} \\ \\ Sinon \\ \\ \quad N.seq, \text{si N.type=f} \\ \\ \quad \oplus_k(M_1.items \ldots M_i.items), \text{ si N.type=}\oplus \\ \\ \quad \cup(M_1.items \ldots M_i.items), \text{ si N.type=}\cup \\ \\ \quad \oplus_0(M_1.items \ldots M_i.items), \text{ si N.type=*, et N.age=0} \\ \quad \oplus_{age}(N.items, N.items), \text{ si N.type=*, et N.age} > 0 \end{cases} \tag{4.1}$$

Données en entrée : E , DB, minsup
(ξ_{th} et ξ_{exp} sont définies au chapitre 4.3)

nr.	Produit la liste des séquences fréquentes satisfaisant la contrainte E
01	Expr = CanonicalForm(E)
02	T = BuildTree(Expr)
03	calculer ξ_{th} pour tout $N \in T$
04	SimplifyTree(T)
05	N.exploration = enAttente pour tout $N \in T$
06	N.etat = Satisfait pour tout $N \in T$ et N.type=*
07	k = 1
08	Ψ_1 = initPhrase(T)
09	C = genCand (Ψ_1)
10	while $C <> \emptyset$ et $\Psi_k <> \emptyset$ do
11	{
12	comptage des candidats
13	séquences fréquentes stockés dans N.items
14	updateNodes(Ψ_k)
15	calculer ξ_{exp} et ξ_{th} pour $N \in \Psi_k$
16	calculer $N.methode$ pour $N \in \Psi_{k+1}$
17	SimplifyTree(T)
18	Ψ_{k+1} = reBuild(Ψ_k)
19	C = genCand(Ψ_{k+1})
20	k++
21	}
22	retuorne T.root.items

TAB. 4.4 – *Pseudo Code de Re-Hackle.*

Les lignes 01 à 08 initialisent les structures de données de l'extraction : l'arbre Hackle (T) correspondant à la forme canonique de la contrainte exprimée par une expression régulière (E) est construit (02), et la première phrase d'extraction collecte toutes les

feuilles de l'arbre (08). La boucle d'extraction (11-21), similaire à celle d'Apriori, opère sur l'arbre Hackle. Elle parcourt et met à jour la phrase d'extraction pour chaque génération de candidats. Les candidats sont générés par les nœuds de la phrase d'extraction (09, 19). Les candidats sont comptés (12-13) à chaque itération, et l'arbre est transformé en fonction de ce qui a été trouvé. La fonction updateNodes (14) effectue les élagages possible après le comptage des candidats. La connaissance de la cardinalité expérimentale et théorique (15) des nœuds permet de choisir une méthode d'extraction localement adaptée à la contrainte (16) et sert lors des optimisations (cf. section suivante). La phrase d'extraction est mise à jour (18) afin qu'elle contienne la liste des nœuds de la génération suivante. La boucle d'extraction (12-20) est répétée tant qu'il y a des candidats à compter ou que la phrase d'extraction contient des nœuds à considérer. Il faut aussi mentionner l'existence de quelques transformations d'arbres (07,17), décrites dans la section optimisations. Enfin, la dernière ligne (22) retourne à l'appelant les séquences fréquentes.

Pour générer les candidats, on utilise actuellement la recopie simple (nœuds union) ou avec comptage (feuilles terminales), une variante d'Apriori pour les fermetures de Kleene (voir la gestion des fermetures de Kleene) et la juxtaposition K-télescopée (pour les concaténations). Les procédures de haut niveau de l'algorithme principal sont expliquées dans la suite.

CanonicalForm transforme une expression régulière dans sa forme canonique. Cette procédure distribue les parties des concaténations de cardinalité théorique unitaire sur les unions. Par exemple, $(a \cup b)c$ devient $(ac \cup bc)$.

BuildExpTree construit l'arbre Re-Hackle correspondant à la contrainte d'extraction. Le pseudo code (récursif) de la procédure de construction est présenté dans la table 4.5.

nr.	
1	HackleTree **BuildExpTree**(Expression E)
2	{
3	si E est une séquence
4	création d'une feuille N
5	sinon
6	décomposition $E = OpPrioritaire(E_1, E_2 \dots E_n)$
7	création du nœud N type OpPrioritaire
8	N.fils[i] = BuildExpTree (Expression E_i)
9	retourne (N)
10	}

TAB. 4.5 – *Construction de l'arbre Hackle.*

InitPhrase initialise la phrase d'extraction initiale Ψ_1 avec les feuilles de l'arbre Hackle listées par un parcours in-order de l'arbre. (Cet ordre n'est pas important, il suffit seulement de parcourir complètement l'arbre.)

GenCand génère les candidats de la présente génération. Les fonctions extractrices sont appelées sur chaque nœud de la phrase d'extraction. Les candidats envoyés par les nœuds sont collectés dans une structure globale pour qu'ils soient comptés durant la même lecture de la BD. Au niveau de l'implémentation, la structure des candidats garde l'information de provenance des candidats afin que les nœuds puissent être mis à jour

après le comptage pour changer leurs états : *Violé* ou *Satisfait*.

Le cas spécial de la fermeture Kleene est discuté plus loin.

```
nr.
 1   Candidats GenCand(ExtactionPhrase Ψ)
 2   {
 3     C = ∅ // C contiendra les candidats
 4     pour tout N_i ∈ Ψ do
 5       C = C ∪ N_i.FonctionGeneratrice
 6     retourne (C)
 7   }
```

TAB. 4.6 – *Génération des candidats par Re-Hackle*

UpdateNodes assure la mise à jour des nœuds appartenant à la phrase d'extraction et de leurs parents. Elle est une étape très importante de l'algorithme et a lieu juste après le comptage des candidats. Les nœuds qui ne génèrent aucun motif sont marqués *Violé* et la partie invalidée de l'expression de départ est écartée de l'arbre Hackle. On examine également les parents du nœud violé qui sont invalidés en cascade jusqu'à ce que leur type ne soit plus une concaténation. Ce fonctionnement a été illustré par l'exemple d'exécution de Re-Hackle.

```
nr.
 1   UpdateNodes(ExtactionPhrase Ψ)
 2   {
 3     pout tout N_i ∈ Ψ do
 4     si (N_i.type != *)
 5     {
 6       N_i.extraction = enCours
 7       si (N_i.items = ∅ )
 8       {
 9         N_i.etat = Violé
10         PropagateViolation(N_i.parent)
11       } sinon N_i.etat = Satisfait ;
13     }
14   si (aucun candidat fréquent lors du dernier comptage)
15     N_i.extraction = finie
16   }
```

TAB. 4.7 – *L'arbre Hackle est mis à jour à chaque génération. La routine Propagate-Violation(Noeud N) est très simple : { si N.type=⊕ N.etat=Viole et PropagateViolation(N.parent) sinon rien}*

ReBuild assure la reconstruction de la phrase d'extraction à chaque génération. Les nœuds, dont l'exploration est terminée sont remplacés par leur parent si :

– tous les descendants du parents sont explorés ;

– le nœud exploré ne se trouve pas sur une branche violée de l'arbre.

La phrase d'extraction doit être renouvelée à chaque itération. Les nœuds dont l'exploration est finie sont enlevés de la phrase et remplacés par leur parent immédiat si tous les fils de ce parent ont été explorés. Seules les fermetures de Kleene peuvent rester plusieurs générations dans la phrase, les autres nœuds sont enlevés au début de la génération suivante. La procédure est illustrée dans la table 4.8.

nr.	
1	ExtactionPhrase **reBuild**(ExtactionPhrase Ψ)
2	{
3	$\Psi_2 = \psi$
4	forall $N_i \in \Psi$ do
5	
6	if N_i.exploration $\neq$ finie // indique une fermeture Kleene
7	add N_i to Ψ_2 ; continue
8	if N_i.etat = Satisfait
9	$\Psi_2 = \Psi_2 \cup N_i$.parent si tous les fils de N_i.parent sont explorés et N_i n'est pas sur une branche déjà élaguée.
10	}
11	return Ψ_2
12	}

TAB. 4.8 – *Mise à jour de la phrase d'extraction.*

ComputeMethod calcule la méthode d'extraction alternative pour tous les nœuds de la phrase d'extraction (cf. optimisations).

4.2.1 Gestion des fermetures de Kleene

Définition 5. *La fermeture Kleene d'une expression régulière $\mathcal{E}$ est l'ensemble des séquences fréquentes de la fermeture Kleene dans le sens de l'equation (2.1) des séquences fréquentes représentes par $\mathcal{E}$, i.e,*

$$\mathcal{E}* = \cup_{k=0}^{\infty} \mathcal{E}^k \tag{4.2}$$

avec

$$\mathcal{E}^{k+1} = \{S \in \mathcal{E}^k \oplus_{k-1} \mathcal{E}^k \mid support(S) > minsup\} \text{ et } \mathcal{E}^0 = \epsilon$$

Les nœuds qui correspondent aux fermetures de Kleene récupèrent les séquences fréquentes venant des niveaux inférieurs connectés à ce nœud. Ensuite, le processus de génération des futurs candidats se poursuit dans un cadre Apriori. Les séquences de longueurs inégales doivent être assemblées et comptées pour générer la fermeture de Kleene. Par exemple, si l'on considère les séquences a, cd et bad venant du niveau inférieur, il faut générer les candidats aa, acd, $abad$, cda, $cdcd$, $cdbad$, $bada$, $badcd$ et $badbad$. Pour l'étape suivante, il faut concaténer seulement les séquences qui partagent une séquence commune de la première génération (a, cd ou bad), ce qui pose problème pour une génération à la Apriori. Une représentation qui ne garde aucune trace de la composition des séquences de seconde (nième) génération rend cette superposition pratiquement impossible.

L'alphabet secondaire. Pour chaque séquence atomique de la forme canonique de la contrainte, on définit un symbole nouveau qui est ajouté à l'alphabet initiale. La contrainte est ensuite représentée dans le nouvel alphabet. Par exemple, $a \oplus (ab \cup fe) \oplus dca$ deviendra $a \oplus (\alpha \cup \beta) \oplus \gamma$ avec trois nouveaux symboles $\alpha = ab$, $\beta = fe$ et $\gamma = dca$. La représentation dans ce nouvel alphabet contient un mélange de symboles initiaux et nouveaux. Comme les nouveaux symboles correspondent aux séquences bâties avec l'alphabet initial, on dit que nous avons un alphabet à deux niveaux. L'algorithme Re-Hackle utilise donc un alphabet à deux niveaux.

Une **séquence granulaire** est une séquence qui contient au moins un symbole tiré de l'*alphabet secondaire*. Elle garde l'information de composition de la séquence. Par exemple, l'expression régulière $ab(c \cup d)b$ génère les deux séquences granulaires : αcb et αdb, avec $\alpha = ab$. Les sous-séquences des séquences granulaires sont aussi des séquences granulaires.

Afin de traiter ce problème, nous introduisons la *concaténation hiérarchico-télescopée*. La méthode fonctionne sur deux niveaux hiérarchiques. Les candidats de niveau 2 sont composés des candidats élémentaires de niveau 1 qui sont les séquences de la première génération. Il s'agit d'enrichir artificiellement l'alphabet de départ en y ajoutant des codes correspondant aux séquences recueillies depuis les nœuds inférieurs. Les candidats de niveau 2 (candidats composés) sont des chaînes constituées à partir des éléments de l'alphabet enrichi. Cette représentation laisse figurer le découpage en candidats élémentaires. Chaque séquence composée sera représentée de deux façons différentes. Prenons l'exemple de la chaîne $acdbad$ qui a la forme $(a)(cd)(bad)$ dans sa représentation hiérarchisée. Sa longueur vaut trois (nombre des séquences atomiques ajoutées à l'alphabet) On parle aussi de la représentation aplatie, où l'information de composition de la chaîne est perdue. Sa longueur est le nombre de symboles la composant (ici 6). C'est la représentation utilisée lors du comptage de candidats.

– Le niveau k de Apriori correspond à l'age du nœud.

– Les items fréquents L_1 sont donnés par les fils du nœud "*".
– Les candidats C_2 sont calculés par juxtaposition.
– A partir d'un age supérieur à 1, on procède à la concaténation k-hiérarchico-télescopée.

Par exemple, on donne (L_k- séquences fréquentes au niveau k, C_k - candidats au niveau k) Soit $\alpha = adc$, $\beta = bed$, $\gamma = bp$
age=1 => $L_1 = \{\alpha, \beta, \gamma\}$
age=2 => $C_2 = \{\alpha\alpha, \alpha\beta, \alpha\gamma, \beta\alpha, \beta\beta), \beta\gamma, \gamma\alpha, \gamma\beta, \gamma\gamma\}$ et soit $L_2 = \{\alpha\gamma\beta\alpha\beta\beta\beta\gamma\gamma\alpha\gamma\gamma\}$
age=3 => $C_3 = \{\alpha\gamma\alpha, \alpha\gamma\gamma, \beta\alpha\gamma, \beta\beta\gamma, \beta\beta\alpha, \beta\gamma\alpha, \beta\gamma\gamma, \gamma\alpha\gamma, \gamma\gamma\alpha, \beta\beta\beta\}$; $L_3 = \cdots$

Chaque séquence de niveau 1 est un symbole de l'alphabet enrichi et compte comme un caractère lors de la superposition des candidats du niveau supérieur. Le lecteur se rendra compte du gain résultant de cette méthode par rapport au principe Apriori de base qui ne pourrait pas gérer les superpositions des préfixes et des suffixes de longueur inégale.

4.3 Optimisations

Une contribution importante de notre travail est la caractérisation de la sélectivité des contraintes. Les concepts suivants (cardinalité théorique et expérimentale) serviront pour la caractérisation formelle de la sélectivité.

La **cardinalité théorique** d'une *sous-expression* est le nombre des *séquences atomiques* qu'elle peut engendrer après l'expansion de tous les *opérateurs*. Les cardinalités théoriques sont mises à jour si la cardinalité des nœuds subordonnés (situés dans le sous-arbre du nœud) change.

La **cardinalité expérimentale** d'une *sous-expression* (et du nœud correspondant) est le nombre des *séquences atomiques* effectivement extraites par ce nœud. Si la cardinalité théorique tient compte uniquement de la sous-contrainte correspondante, la cardinalité expérimentale correspond à la réalité observée dans la base de données des séquences après le comptage des candidats.

Les cardinalités sont définies par l'equation (4.3) et l'equation (4.4).

$$\xi_{th} = \begin{cases} N.\xi_{exp} \text{ , si N.Exploration = finie} \\ \text{sinon} \\ \quad 1 \text{ , si N.type} = f \\ \quad \prod_{M_i \in N.fils} M_i.\xi_{th} \text{ , si N.type} = \oplus \\ \quad \sum_{M_i \in N.fils} M_i.\xi_{th} \text{ , si N.type} = \cup \\ \quad Card^3(N.fils.items) \text{ , si N.type= *} \end{cases} \tag{4.3}$$

Il est difficile de donner une borne supérieur au nombre d'éléments retournés par les fermetures Kleene, car le langage est infinie. Cependant, nous avons trouvé expérimentalement que le cube des séquences fréquentes dans le fils de la fermeture donnait de bons résultats à l'exécution. Il est possible de fournir une formule plus sophistiquée, qui se met à jour lors des évaluations successives de la fermeture - sans aucun gain apparent lors de nos tests.

Par exemple, au début de l'exécution de Re-Hackle, les n œuds 11 et 5 de la figure 4.2 ont la cardianlité théorique 4 et respectivement 5, alors que celle de la racine est 10.

$$\xi_{exp} = \begin{cases} Card(N.items) \text{ si N.Exploration = finie} \\ \text{sinon} \\ \quad \prod_{M_i \in N.fils} M_i.\xi_{exp} \text{ si N.type} = \oplus \\ \quad \sum_{M_i \in N.fils} M_i.\xi_{exp} \text{ si N.type} = \cup \\ \quad Card(N.fils.items) \times Card(N.items) \text{ si N.type} = * \end{cases} \tag{4.4}$$

La cardinalité théorique est recalculée à chaque génération en fonction des candidats trouvés aux niveaux inférieurs tant que l'exploration du nœud n'est pas achevée. Elle devient alors le nombre des séquences associées à ce nœud. La cardinalité théorique converge (au cours des itérations) vers le nombre des séquences fréquentes dans chaque nœud. A chaque moment les cardinalités sont des estimations des bornes supérieure et inférieure du nombre des séquences fréquentes de chaque nœud, sans être des vrais bornes au sens mathématique.

Il existe un certain nombre d'optimisations possibles qui améliorent de manière spectaculaire les performances de l'algorithme. L'analyse du comportement de notre méthode nous a permis de voir les cas ou les performances de l'algorithme peuvent être améliorées :

- Il faut éviter l'explosion des candidats pour les concaténation de très grande cardinalité.
- Donner une garantie que les fermetures de Kleene n'augmentent pas le nombre des lectures de la base des séquences.
- Chercher à assembler des séquences de plus en plus longues afin de bénéficier des accélérations dues à des granularités élevées.

4.3.1 Simplification de l'arbre

La simplification a été introduite afin de compacter l'arbre correspondant à l'expression de départ. Un arbre plus compact génèrera moins de candidats et aura besoin de moins de passes sur la BD. L'intuition est qu'une expression de type $a(b \cup c)$ pourra être traitée directement comme deux séquences ab et ac au lieu de tester séparément a, b, c et ensuite ab, ac. La génération directe des candidats de longueur 2 est justifiée par le fait que l'éventuel élagage au niveau 1 (si l'un de a, b, c était infréquent) ne diminuerait pas le nombre de candidats ou les passes sur la BD. Il faut repérer les cas où l'élagage au niveau précédant ne réduira pas le nombre de candidats et faire l'hypothèse que les séquences du niveau suivant seront fréquentes. Si c'est le cas et que des séquences longues sont fréquentes, alors le nombre global des candidats diminue. Dans le cas contraire, l'algorithme peut se rattraper au niveau suivant, où les candidats à compter seront certes plus longs, mais leur nombre restera inférieur ou égal au nombre de candidats de la méthode de base.

Afin de simplifier l'arbre, on repère les nœuds de cardinalité expérimentale (théorique, si l'extraction n'a pas encore débuté) unitaire, et on les annexe à leurs voisins gauche et droite.

Le premier fils d'un nœud représentant une concaténation est appelé **orphelin préfixe** si sa cardinalité expérimentale (ou théorique, avant l'extraction) vaut 1. Le dernier fils d'un nœud est appelé **orphelin suffixe** si sa cardinalité expérimentale vaut 1. Les orphelins préfixes sont intégrés à l'arbre codifié par leurs voisins à droite, les orphelins suffixes sur le

sous-arbre qui les précède, et ensuite enlevés de l'arbre d'expression. Ce processus s'appelle *distribution d'un orphelin sur son voisin.*

Théorème 2. *La distribution d'un orphelin sur son voisin ne modifie pas le nombre de candidats générés par ce dernier. De plus, le comptage des candidats complétés par un orphelin n'induit aucun surcoût par rapport au cas du comptage sans gestion des orphelins.*

Démonstration. La séquence codifiée par l'orphelin s'ajoute comme préfixe (resp. suffixe) sur tout candidat généré par le nœud voisin et l'orphelin ne peut pas engendrer de séquence fréquente satisfaisant la contrainte de son parent tout seul. Prenons d'abord le cas des orphelins préfixes.

Soient P la séquence préfixe de l'orphelin préfixe et $\{X, Y, \ldots\}$ les séquences candidates du nœud successeur. Si P est fréquent, on compte seulement $PX, PY...$ au lieu de compter séparément $P, X, Z..$ et ensuite $PX, PY....$ Le nombre de candidats est divisé par deux ! Si P n'est pas fréquent, on compte $PX, PY..$, au lieu de $P, X, Z..$, ce qui peut être implémenté de façon très efficace.

Le même raisonnement s'applique aux orphelins suffixes et au suffixe S.

4.3.2 Flux ascendant des candidats

Considérons maintenant une expression régulière de la forme $a * b$, et une BD avec de longues séquence de a mais aucun ab. Toutes les a seraient comptés, ce qui n'est pas efficace. Pire, la présence d'une fermeture de Kleene peut retarder l'évaluation de la séquence fréquente d'une passe. On peut donc trouver des cas particulièrement désastreux, comme $((a * b) * c) * d$ qui aurait besoin au minimum de 7 passes sur la BD avant de produire la séquence $abcd$, supposant que la deuxième évaluation de chaque fermeture est l'ensemble vide ($\emptyset$). Dans ce cas, on aurait les candidats suivants à chaque génération : 1 : $\{a, b, c, d\}$, 2 : $\{aa\}$, 3 : $\{ab\}$, 4 : $\{abab\}$, 5 : $\{abc\}$, 6 : $\{abcabc\}$ et 7 : $\{abcd\}$ (d'autre candidats apparaitront a partir de la génération 4, parce que le fermetures Kleene contiennent toujours la chaine vide, ϵ). Grâce a cette exemple, il est facile de réaliser que chaque niveau d'imbrication d'une fermeture Kleene évaluée induit une nouvelle passe sur la BD. Bien que cette génération supplémentaire ne soit pas très pénalisante pour l'ensemble de l'extraction (en attendant la fin de l'exploration de la fermeture, l'algorithme pourra travailler dans les autres branches de l'arbre), il est préférable d'avoir la garantie que les structures d'imbrication complexes ne pénaliseront pas l'exécution. Pour ce faire, nous introduisons un mécanisme de réutilisation des séquences fréquentes dès leur génération dans une fermeture de Kleene, et sans attendre que toutes les séquences de la fermeture soient extraites. Les séquences fréquentes peuvent remonter aux niveaux supérieurs dès qu'elles sont identifiées.

L'hypothèse initiale que la phrase d'extraction ne contient jamais de nœuds en relation parent-fils doit être relaxée. La procédure qui reconstruit cette phrase à chaque génération devient plus complexe tout comme la propagation des violations et la génération des candidats.

Si le parent d'une fermeture est une concaténation et si la première génération des séquences concaténées est vide, on peut arrêter l'exploration de la fermeture, car elle n'apportera aucune séquence fréquente au niveau supérieur. En procédant ainsi, la seconde génération ($k = 2$) de la fermeture et les concaténations de la première génération avec les

autres fils sont comptés en même temps. De manière générale, lors du calcul de la $k$ième étape de fermeture, les séquences de niveau $k-1$ sont systématiquement concaténées aux autres frères de la fermeture. L'exemple précédent se résout en quatre passes seulement : $1 : \{a, b, c, d\}$, $2 : \{aa, ab\}$, $3 : \{abab, abc\}$, $4 : \{abcabc, abcd\}$.

Le tableau 4.9 compare l'algorithme de base (sans le flux ascendant) à la variante optimisée. Soit la contrainte $a * b*$, et supposons que la BD contienne *aaaacbb* comme séquence fréquente. Le flux ascendant reconnaît qu'il n'y a pas de séquence *ab* dans la base, et ne comptera pas toutes les séquences composées de *a*. Ici, il est donc possible d'économiser quatre passes sur la BD.

Passe	Hackle (base)	Flux ascendant
1	**a, b**	**a, b**
2	**aa**, ab	**aa**, ab
3	**aaa**	Stop
4	**aaaa**	
5	aaaaa	
6	ab, aab, aaab, aaaab	
7	Stop	

TAB. 4.9 – *Flux ascendant de candidats. Les symboles en gras sont fréquents.*

Notons, que le flux ascendant des candidats est une méthode locale qui gère les concaténations à gauche et à droite, ce qui est un critère d'élagage (à base de fréquence) bien plus général que celui de Spirit(V), qui ne pourrait élaguer la génération des *a* si la contrainte était $ba*$. Re-Hackle repère ce cas et procède à l'élagage correspondant.

Cette optimisation n'est pas simple techniquement. La version initiale de notre algorithme Re-Hackle contenait des branchements conditionnelles d'une grande complexité. Heureusement, le cadre CCM présenté au chapitre 3, permet de reformuler cette logique avec des opérations d'arbres : on utilise la duplication pour faire remonter les candidats au niveau supérieur avec la même logique de reconstruction de la phrase d'expression sans fermeture Kleene, alors que la fermeture survit dans un arbre parallèle.

4.3.3 Adaptation de la méthode d'extraction

Quand certaines sous-expressions adjacentes sont déjà évaluées et que les autres sous-arbres demandent encore beaucoup de traitement, l'algorithme peut décider de regrouper deux par deux les fils adjacents explorés pour créer de nouveaux nœuds. Le déclenchement du processus est piloté par la cardinalité des fils.

L'explosion du nombre de candidats lors des concaténations est un problème majeur de l'algorithme de base. Prenons, par exemple la concaténation d'une dizaine de nœuds chacun renvoyant de 3 à 4 candidats. La méthode de génération traditionnelle donnerait entre 59.049 et 1.048.576 candidats, ce qui n'est pas pratique. On assiste à une explosion combinatoire qui sort vite du domaine du traitable.

La solution est donnée par la technique de l'**adaptation de la méthode d'extraction**. Dans certains cas, il est possible de regrouper les nœuds deux par deux, construire

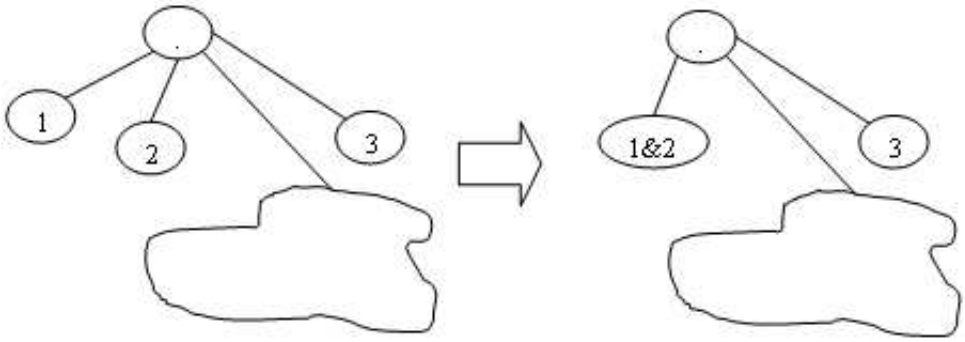


FIG. 4.4 – *Adaptation par combinaison de deux nœuds.*

plusieurs niveaux hiérarchiques et se placer dans un contexte de superposition des candidats pour davantage d'élagage à base de fréquence. La figure 4.4 illustre ce principe.

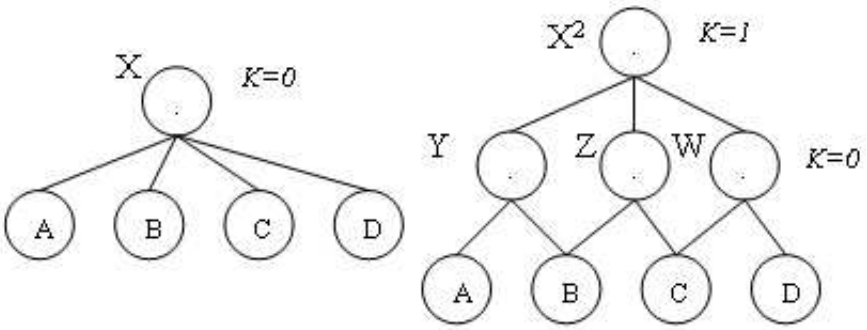


FIG. 4.5 – *Adaptation par insertion d'un niveau supplémentaire*

Supposons que les fils A, B, C, D de S sur la figure 4.5, contiennent trop de séquences candidates. Dans ce cas, la cardinalité théorique de X devient dangereusement élevée, et il est plus avantageux de remplacer X par un nœud X^2 qui a un étage en plus. Les fils initiaux sont regroupés deux par deux et donnent Y, Z, W afin de procéder aux élagages. Comme les séquences de Y ont comme suffixe les séquences de B qui sont également les préfixes de Z, les candidats de X^2 seront constitués par superposition k-téléscopée (avec k=1) des éléments de Y, Z, W. Après l'opération, X^2 contient trois fils avec un nombre de séquences candidates probablement inférieur à celui de X. Si ce chiffre est encore trop élevé, l'algorithme pourra décider d'introduire encore un niveau X^3 (cette fois avec k=1 et k=2). Le nombre des niveaux insérés est déterminé en cours de l'exécution. L'introduction des niveaux multiples augmente le nombre de passes de la base de séquences, mais diminue le nombre des candidats. Des candidats ont été troqués contre des passes sur la BD supplémentaires, mais notons, que la longueur des séquences augmente à chaque niveau de l'arbre, ce qui garantit que l'algorithme ne fera pas plus de passes que GSP ou Spirit(L et V). Le mécanisme de pilotage de la réduction du nombre des candidats en introduisant des passes BD est assez complexe et dépend de la taille de la BD, du coût d'accès aux données, et de la cardinalité expérimentale des fils du nœud de concaténation.

4.4 Validation expérimentale

L'algorithme Re-Hackle a été testé sur deux types de jeux de données : un jeu artificiel suivant la distribution Zipfien (http ://linkage.rockefeller.edu/wli/zipf/), et des données réelles provenant des fichiers logs d'un site web. Cette distribution Zipfienne est une modélisation des probabilités d'occurrences inspirée des langues naturelles dans laquelle le $k$ième symbole dans l'ordre décroissant des fréquences apparait avec une probabilité de :

$$p_k = \frac{k^{-a}}{\sum_{i=1}^{n} i^a} \text{ , avec } a \text{ proche de 1} \tag{4.5}$$

4.4.1 Densité des données

La densité des données a été simulé en diminuant le nombre de symboles de notre distribution zipfien, les autres paramètres restant les mêmes. Le temps d'exécution de RE-Hackle n'augmente pas de façon exponentielle avec la densité des données (comme c'est la cas de Spirit(N) et en moindre degré de Spirit(L)).

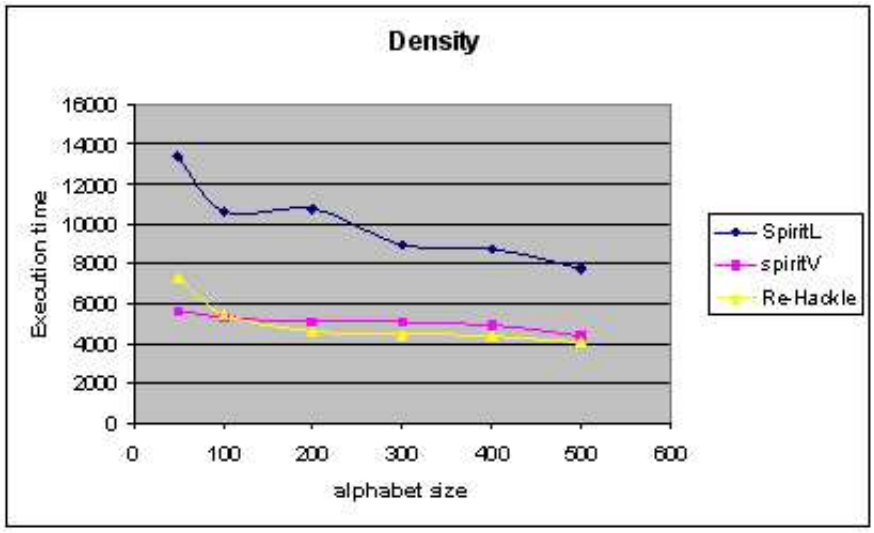


FIG. 4.6 – *Bon comportement vis-à-vis de l'augmentation de la densité des données, pas d'explosion exponentielle.*

Selon la figure 4.6, plus les données sont denses, mieux Re-Hackle se démarque des approches traditionnelles pour lesquels on observe une croissance exponentielle du temps d'exécution.

4.4.2 Granularité de la contrainte

Définissons la *granularité* comme le ratio entre le nombre des littéraux de l'expression et nombre des séquences atomiques. Les exemples suivantes donnent quelques exemples. $ab.(cde|fg).cf$ [9/4=2.25] , $ac|d.(g|bb)e$ [7/5=1.4] ou $(a|b|c) * d(t|z)$ [6/6=1], etc. Cette valeur et toujours un nombre inclus entre 1 et le nombre des littéraux.

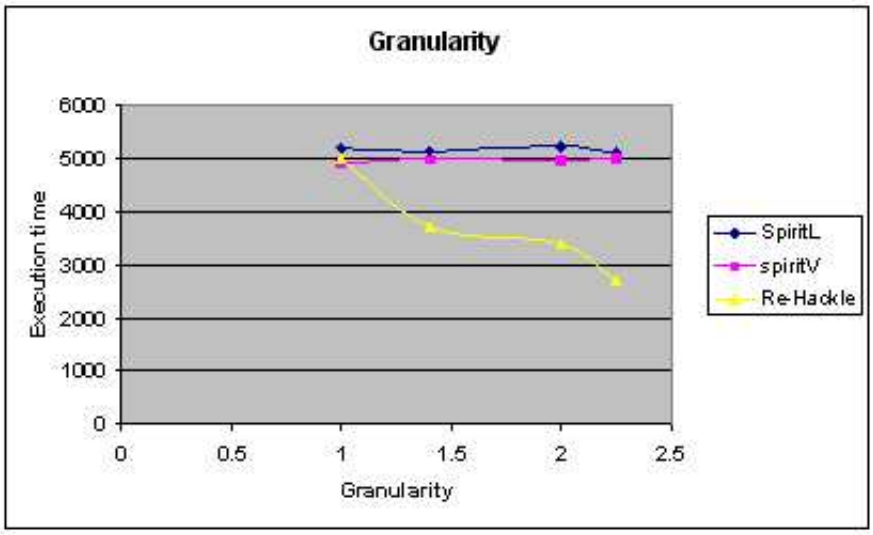


FIG. 4.7 – *Dépendance du temps d'exécution de la granularité de la contrainte*

Selon la figure 4.7, le temps d'exécution diminue si la granularité de la contrainte augmente. La nature de la dépendance est encore à étudier. Les algorithmes Spirit(N, L et V) sont pratiquement insensibles à ce facteur.

4.4.3 Passage à l'échelle

Le comportement de l'algorithme reste linéaire si on augmente le nombre de séquences dans la BD ou la longueur des transactions (voir figure 4.8).

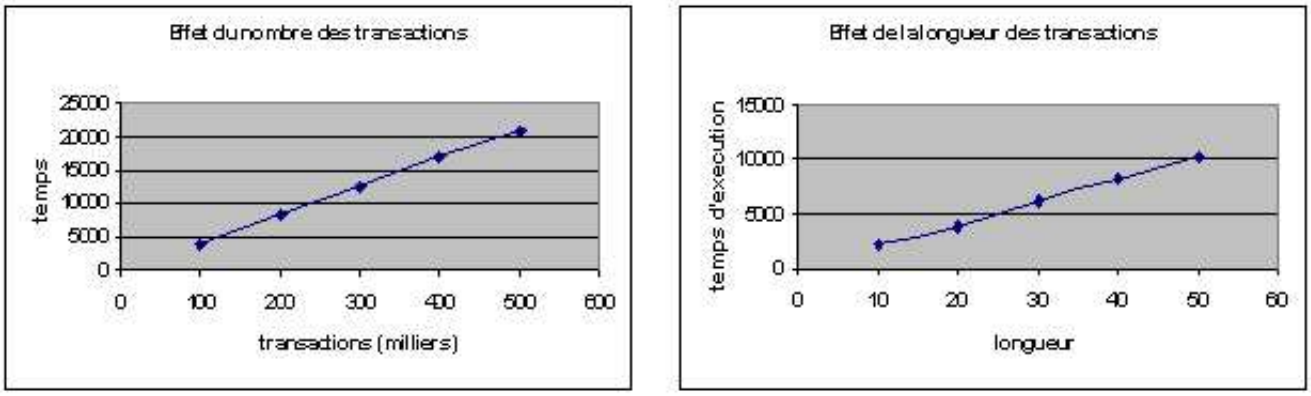


FIG. 4.8 – *Dépendance du temps d'exécution de la granularité de la contrainte*

4.4.4 Comparaison à la famille Spirit

La force de la stratégie d'élagage de Re-Hackle est entre celle de Spirit(L) et Spirit(R). Comme Spirit(L) fonctionne principalement avec un élagage à base de fréquence, et Spirit(R) s'appuie sur l'élagage basé sur l'expression régulière, Re-Hackle ne surpassera pas toujours toutes les deux variantes. Néanmoins, il peut être plus rapide que le meilleur Spirit, car son élagage est basé sur la conjonction des deux critères. L'atout majeure de Re-Hackle est d'avoir un comportement stable : s'il est impossible de savoir avant l'exécution laquelle variante de Spirit marcherait le mieux pour un jeu de données, Re-Hackle a bon comportement en moyenne.

Les résultats expérimentaux de la figure 4.9 reflètent le fait que Re-Hackle prend la forme de la meilleure variante Spirit et adopte la meilleure stratégie d'élagage calculée en fonction de la cardinalité de la contrainte (et donc de la base de séquences).

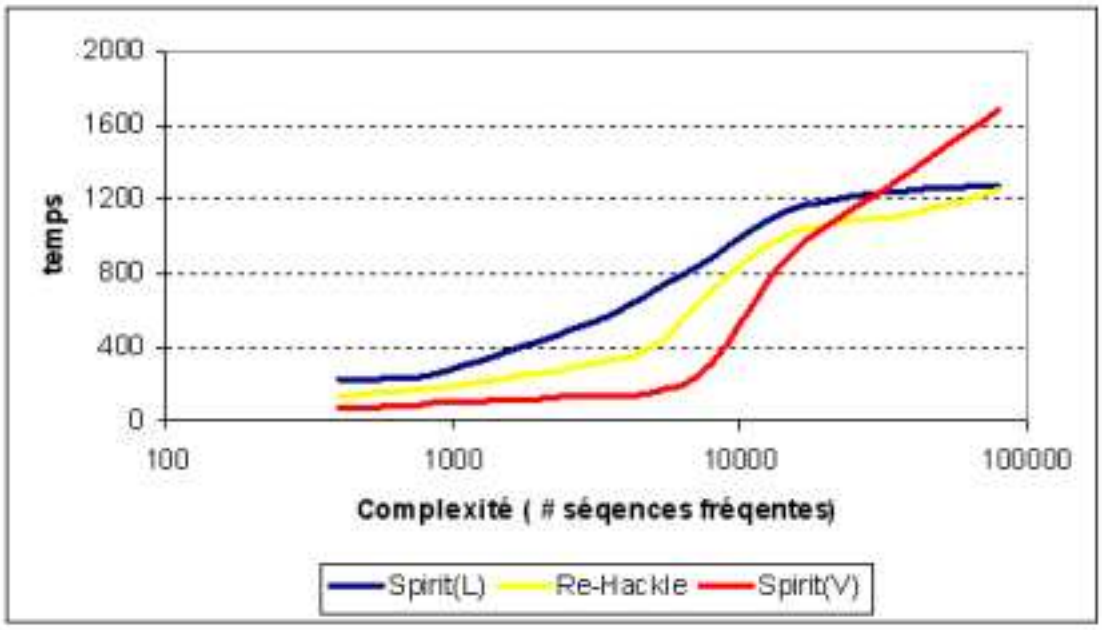


FIG. 4.9 – *Re-Hackle adopte la stratégie de la meilleure variante Spirit.*

4.4.5 Effet des optimisations

Les optimisations introduites sont très importantes. Regardons une concaténation de six sous-expressions, contenant toutes six séquences fréquentes. Les observations sont résumées dans le tableau 4.10.

Il apparait clairement que faire trois passes additionnelles sur la BD afin de bénéficier de plus d'élagage à base de fréquence réduit considérablement le nombre de candidats, avantage que l'on retrouve dans le temps d'exécution divisé par deux. L'emploi d'une stratégie adaptative est amplement justifié. Notre second exemple montre que le flux de candidats ascendant réduit le nombre de passes . L'exemple du tableau 4.11 a été calculé

Critère	Hackle optimisé	Hackle (base)
Temps d'exécution	127	269
Candidats	139	21617
Passes DB	5	2

TAB. 4.10 – *Amélioration due à l'adaptation de la stratégie d'élagage.*

Critère	Hackle optimisé	Hackle (base)
Temps d'exécution	21	32
Candidats	51	51
Passes DB	6	9

TAB. 4.11 – *Amélioration due au flux ascendant.*

pour la contrainte $((a * b) * c*)d$ avec *aabcabcd* comme seule séquence fréquente.

Dans ce cas simple le nombre des candidats est faible, et le temps d'exécution est déterminé par le nombre de passes sur la BD. Le flux ascendant ne réduit pas le nombre de candidats comptés, mais accélère l'assemblage des séquences finales et contribue à une extraction plus rapide.

4.5 Conclusion

L'introduction d'un nouveau cadre d'extraction permettant la prise en compte de propriétés locales est une voie à haut potentiel. Nous avons contribue des 2003 a l'extraction sous contraintes en proposant la technique Re-Hackle qui représente les contraintes par des arbres abstrait de syntaxe. Dans ce cas concret des contraintes exprimées par expressions régulières, l'algorithme exploite activement les cardinalités des sous-expressions pour bénéficier à la fois de l'élagage à base de fréquence et selon la contrainte d'expression régulière. Re-Hackle propose un meilleur compromis entre ces deux critères de réduction de l'espace de recherche que le cadre Spirit. La granularité de la contrainte et l'emploi des techniques d'extraction choisies en fonction des propriétés locales de la contrainte font que les deux critères d'élagage peuvent bien coexister. La méthode d'extraction automatiquement détectée par l'algorithme en fonction de la cardinalité expérimentale des nœuds et de la contrainte imposée permet le choix de la technique localement adaptée à chaque partie de l'extraction. Le choix de l'introduction des niveaux supérieurs dans l'arbre se fait en tenant compte du nombre des candidats. A notre connaissance, l'injection de cette information dans le traitement de la sous-partie d'une expression est original. De fait, notre travail a été l'une des premières approches proposées pour des algorithmes de fouille de données ayant des stratégies d'élagage adaptatives.

Le cadre théorique est bien fondé, et les résultats expérimentaux montrent que nous avons fait un progrès vers l'intégration active au sein de l'extraction des contraintes qui n'ont pas la propriété d'anti-monotonie dans l'espace de recherche initial. Nous donnons une réponse possible aux problèmes soulevés par les algorithmes Spirit : le degré de re-

laxation des contraintes afin de bénéficier à la fois des élagages à base de fréquence et selon la contrainte ER. De plus, la technique du flux ascendant des candidats ramène les expressions régulières dans un cadre presque aussi fort que les contraintes anti-monotones.

Deuxième partie

Décisions financières

Introduction

Une grande partie de la thèse s'est déroulée dans des établissements bancaires. Ce contexte est bien particulier quant à la formation et aux besoins des intervenants. C'est un environnement de recherche stimulant posant de nombreux défis originaux.

Une banque interagit avec les bourses via sa salle des marchés ("trading floor"). Cette structure réunit les vendeurs qui connecte la banque à ces clients aux investisseurs institutionnels et entreprises et les traders qui décident les ordres de bourse les mieux adaptés aux intérêts de la banque. Cette thèse est affiliée à deux types d'activité de trading : (a) d'abord à l'activité de trading au compte propre qui investit l'argent de la banque et cherche des outils de prédiction des cours des actifs financiers, et (b) ensuite aux produits structurés. Cette dernière conçoit des produits financiers complexes adaptés aux besoins du client, et gère les risques associés. Elle a besoin de connaître la valeur des produits vendus, et de simuler l'évolution stochastique des flux d'argent des produits commercialisés.

Le trading au compte propre repose d'une part sur la mémoire du trader de motifs vus dans le passé qui ont tendance de se répéter de façon similaire, et à l'analyse de la position macro-économique qui va déterminer les cours, d'autre part. Grâce aux capacités intellectuelles du trader au compte propre, qui identifie - à tort ou à raison - certains mécanismes de marché, les prises de positions au compte propre devraient produire un rendement d'espérance positive. Réalisons, que les qualités d'un bon trader correspondent aux objectifs que se fixent les méthodes de "data mining". Il s'agit de traiter l'information disponible de façon intelligente et d'identifier les motifs invisibles à l'œil profane : le trader "initié" à la bourse voit les mécanismes cachés et peut en tirer profit. Le chapitre suivant cherche à améliorer les outils d'analyse des traders. Les *indicateurs techniques* sont des signaux d'achat et de vente qui aident le trader dans sa décision. Ainsi, notre *première application* (chapitre 5) ajoute une composante de mémoire à ces indicateurs techniques, pour que les signaux d'achat et de vente soit envoyé uniquement si le même indicateur technique a "eu raison" dans le passé pour un contexte similaire. Sur un plan technique, cette application exploite astucieusement la technique des ensembles fréquents dans la recherche de motifs en deux dimensions (cinq flux discrets sur N jours).

L'activité de structuration fait appel à la notion du produit dérivé . Un produit dérivé est un actif financier dont la valeur dépend de la réalisation d'un autre actif. Un exemple simple serait l'option d'achat européenne (call), qui paye la différence entre le niveau d'un actif au moment T (maturité de l'option) et un niveau prédéfini (strike de l'option) si cette différence est positive, sinon rien, i.e., $\max(S(T) - K, 0)$. La valeur de l'option dépend de la valeur du sous-jacent à la maturité, donc il s'agit d'un produit dérivé.

Afin d'illustrer l'importance de notre *seconde application* (chapitre 6.3), considérons

un produit financier sur 5 ans ou chaque année l'investisseur reçoit la différence entre le cours de deux actions (S_1, S_2) si celle-la est positive et paye K unités monétaires, i.e. $\max(S_1(T_i) - S_2(T_i), 0) - K$. Notons $X_i = S_1(T_i) - S_2(T_i)$, et $x^+ = \max(x, 0)$. Les sommes échangées à chaque date sont une variable aléatoire (positive ou négative), comme la figure 4.10 l'illustre.

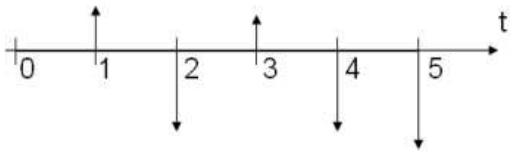


FIG. 4.10 – *Une réalisation des flux de la banque pour le contrat* $X_i^+ - K$.

En supposant que les taux d'intérêts sont nuls, on peut écrire que la valeur du contrat est l'espérance des futurs flux. Celle-ci peut être calculée par une seule simulation Monte Carlo :

$$V = E\left(\sum_{i=1}^{5} X_i^+ - K\right) \tag{4.6}$$

Si on complique maintenant le contrat, en ajoutant une option bermuda après le troisième payement, la valeur du produit baisse pour l'investisseur. Effectivement, la banque arrêtera le contrat, si l'espérance des échanges devient positif pour l'investisseur, qui pourra s'attendre à ce que la banque rappelle le produit si à la fin de la troisième année l'espérance des deux dernier flux devient positif pour lui. Dans ce cas, la valeur du contrat pour l'investisseur devient :

$$V = E\left[\left(\sum_{i=1}^{3} X_i^+ - K\right) + E\left(\min\left(\sum_{i=4}^{5} X_i^+ - K, 0\right)\right)\right] \tag{4.7}$$

Il faut réaliser, que l'espérance imbriquée de l'équation précédente est conditionnelle, car x_4 et x_5 dépendront de x_1, x_2, x_3.

La forme de l'equation (4.6) était $E(f(X))$, alors que l'equation (4.7) contient une espérance double de forme $E(Y + E(f(X) \mid Y))$ (en général, $E(f(X) \mid Y) \neq E(f(X))$). Il est extrêmement coûteux de calculer directement ces espérances imbriquées, car X dépend de l'instance de Y.

La contribution de neighbourhood Monte Carlo proposé dans le chapitre 6.3 consiste à approximer les espérances qui s'écrivent sous la forme $E(Y + E(f(X) \mid Y)))$ avec un seul MC de la forme de l'equation (4.7).

Chapitre 5

Amélioration des Indicateurs Techniques

Les indicateurs techniques sont des fonctions des données de marché, historiques et actuelles, qui produisent un signal d'achat ou de vente. Ce sont les *briques* qui permettent de construire des stratégies de trading en réaction aux indicateurs et en fonction de la composition du portefeuille de l'investisseur. Certaines techniques de fouille de données permettent d'attribuer une signature aux configurations de marché précédant le déclenchement d'un indicateur technique. La comparaison des performances de l'indicateur seul et du même indicateur précédé d'une signature permet de choisir les signatures qui améliorent les performances de l'indicateur. La stratégie de trading peut donc exploiter un filtrage pour se restreindre aux transactions qui se déclenchent après une signature et qui ont produit des gains dans le passé. Les signatures sont testées sur une période d'apprentissage afin d'exclure celles pour lesquelles l'indicateur technique a fait moins bien que l'indicateur non-signé. Les meilleures signatures sont gardées pour la période suivante, dite de validation ou de trading pendant laquelle l'indicateur technique est pris en compte uniquement s'il est précédé par une *bonne* signature. L'analyse des signatures montre suffisamment de stabilité temporelle pour définir un système de trading recalibré période après période. Dans nos tests, l'indicateur filtré par les bonnes signatures de la période d'apprentissage a l'habitude de surpasser l'indicateur seul pendant la période de validation. Nous utilisons une technique de fouille de données classique (i.e., la recherche de motifs fréquents) pour identifier les signatures qui caractérisent les quelques jours précédant le déclenchement de l'indicateur technique. Nous obtenons ainsi des règles d'analyse technique spécifiques au sous-jacent, basées sur plusieurs jours et adaptées à la période considérée. Notre première contribution a donc été de caractériser les configurations de marché d'une façon pertinente pour que l'extraction de motifs fréquents puisse s'appliquer. A notre connaissance, l'idée d'améliorer un indicateur technique par la méthode des signatures est également originale. Nous ne proposons pas un nième indicateur technique mais plutôt une méthodologie générale pour améliorer des stratégies quantitatives existantes. Notons que l'amélioration des indicateurs techniques par les motifs fréquents est un objectif bien différent de la prédiction des cours boursiers et a reçu moins d'attention. F. Allen et R. Karjalainen ont travaillé sur la combinaison et le calibrage des seuils de déclenchement des indicateurs par la programmation génétique [10].

Les travaux plus récents de Backer rapportent que les règles de trading générées par la programmation génétique peuvent battre la stratégie *acheter et garder* en tenant compte des frais de transactions [16]. Ces travaux concernent le paramétrage d'un ensemble de règles de trading déjà connues et ne cherchent pas à caractériser une configuration de marché en addition d'un indicateur technique. Notre approche est donc complémentaire de ces travaux.

5.1 Méthodologie

Pour mettre en place la stratégie de trading basée sur des signatures il faut d'abord définir le contenu d'une configuration de marché (scénario) et le transformer dans le format attendu par l'algorithme d'extraction des motifs utilisé (discrétisation et agrégation).

5.1.1 La description des scénarii

Un scénario regroupe l'ensemble des données de marché historiques sur N jours qui constitue le domaine de l'indicateur. Les indicateurs techniques sont donc des fonctions $F(..)$ des scénarii qui associent une valeur de l'ensemble {Acheter, Vendre, Rien} à un scénario de marché. Par exemple, la fonction F pourrait être de type Acheter, si la moyenne mobile sur deux semaines est supérieure à la moyenne mobile sur 30 jours, sinon Rien. Il s'agit évidemment d'un exemple hypothétique. Comme dans la plupart des travaux dans ce domaine, chaque journée composant un scénario sera caractérisée par la valeur de l'indicateur technique, le cours d'ouverture (O) et de clôture (C) ainsi que le plus haut(H) et le plus bas (L) des cours journaliers. Le nombre de jours N composant un scénario est déterminé empiriquement. De nombreuses techniques de fouille de données travaillent sur des données discrètes et supposent un pré-traitement. Ainsi, nous voulons transformer les données de marché composant les scénarii en une représentation non numérique. L'association de chaque valeur à sa q-quantile apparaît comme un choix de discrétisation simple facilement mis en œuvre. Les deux scénarii quantilisés associés aux journées J_1 et J_2 seront graphiquement représentés comme suit :

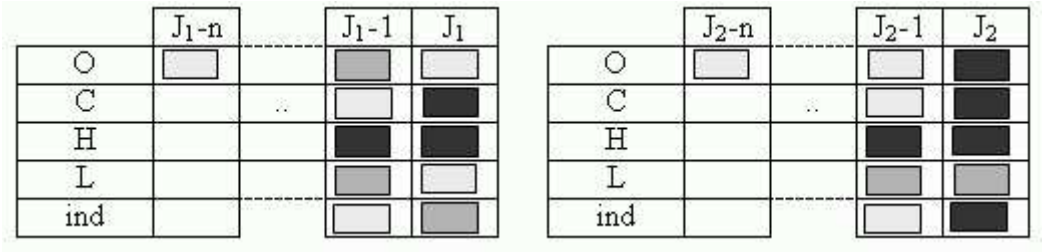


FIG. 5.1 – *Les carrés foncés indiquent des valeurs du tiers supérieur, le semi-foncé représente une valeur moyenne, et le clair une donnée du tiers inférieur.*

La représentation de la figure 5.1 caractérise les niveaux absolus des quantiles. Il est quelquefois intéressant de normaliser cette représentation, par exemple en soustrayant le plus petit quantile des autres quantiles. Cette alternative généralise la représentation précédente et permet l'expression en valeurs relatives. L'intuition derrière cette nouvelle

représentation est de regrouper les configurations similaires à deux niveaux de marché différents (e.g., clôture supérieure d'un quantile à l'ouverture dans un marché bas et un marché haut) et d'en extraire un signal plus fort.

Notre méthodologie associe une signature aux scénarii fréquents en passant par l'extraction de motifs (itemsets) fréquents. L'extraction de motifs fréquents est l'une des tâches de fouille de données qui a été la plus étudiée depuis une dizaine d'années [2, 3, 70, 19, 88]. Cependant, nous n'avons pas traité ce problème dans notre état de l'art puique nous nous concentrons sur la fouille de séquences. Ici, un motif f-fréquent (noté simplement fréquent dans la suite) est un sous-ensemble commun à une collection d'ensembles (scénarii dans le cas présent, collection de transactions dans le cadre des données de ventes) qui apparaît dans au moins f ensembles. L'extraction des motifs fréquents intervient sur des ensembles de valeurs et il faut donc représenter chaque scénario sous une forme transactionnelle. On préfère souvent une représentation suffisamment riche pour pouvoir reconstituer le scénario de départ à partir d'un ensemble non ordonné, i.e., une bijection. La bijection a l'avantage de pouvoir retrouver la configuration de marché à partir de sa signature, et permet à un expert humain de comprendre a postériori les règles générées par la technique de fouille de données. En introduisant un système de coordonnées (x, y) comme sur la figure 5.2, il est possible de définir une bijection pour chaque valeur composant un scénario. Un exemple de transformation est donné par la formule :

$Code = 5 * x * q + y * q + value(x, y)$

où q est le nombre de quantiles et 5 le nombre de données par jour (indicateur, ouverture, clôture, plus haut journalier, plus bas journalier) Par construction des quantiles $value(x, y) < q$.

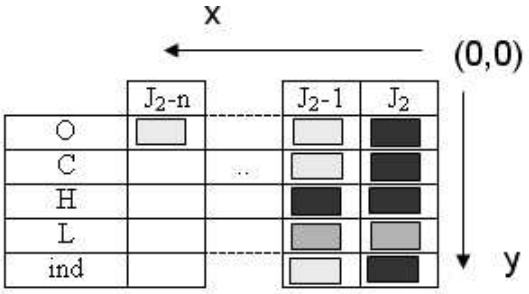


FIG. 5.2 – *Transformation d'une configuration de marché dans une forme ensembliste.*

Pour les deux dernières colonnes des scénarii de la figure 5.1, on obtient les ensembles suivants (q=3) avec explicitation du calcul de la seconde valeur de la colonne J_2-1 ($x = y = 1$) :

$$\begin{aligned} J_1 &= \{1; 6; 9; 10; 14; 17; 19; 24; 26; 28\} \\ J_2 &= \{3; 6; 9; 11; 15; 16; 19 = 3 * 5 * x + 3 * y + 1; 24; 26; 28\} \end{aligned}$$

Après l'étape de discrétisation et de codification, chaque jour est représenté par un ensemble de $5xN$ symboles obtenus par la formule précédente. C'est bien un ensemble et non pas une séquence, l'ordre des symboles n'a plus d'importance.

5.1.2 Les signatures

On souhaite caractériser les scénarii fréquents qui peuvent être à la base d'une stratégie quantitative. Dans le contexte d'amélioration des indicateurs techniques avec notre méthode, il n'est pas possible de signer les événements rares. Il est souhaitable que la fréquence du motif soit comparable à la fréquence de l'indicateur et les motifs fréquents apparaissent donc opportuns. La signature d'un scénario est un sous-ensemble (sous-motif) fréquent (dans les ensembles des scénarii) des motifs qui le décrivent. Les scénarii peu fréquents n'ont pas forcément de signature et un scénario peut avoir plusieurs signatures. Nous donnons un exemple de signature (i.e., un sous-motif commun) des deux scénarii de la Figure 5.1. Le sous-ensemble commun de J_1 et de J_2 est 6 ; 9 ; 20 ; 24 ; 25 ; 29 ce qui, grâce à la bijection, correspond au motif visuel donné dans la Figure 5.3. Notons que les sous-ensembles des signatures sont également des signatures. Dans le contexte de l'amélioration des indicateurs, la période étudiée comprend quelques centaines d'ensembles et les seuils de fréquences absolues utilisés sont de l'ordre de quelques dizaines.

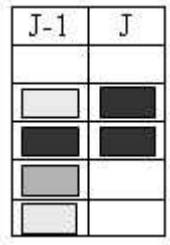


FIG. 5.3 – *Signature des scénarii de la Figure 5.3.*

5.1.3 L'algorithme d'extraction

L'extraction des sous-ensembles d'une collection de scénarii est une tâche difficile car les approches naïves se heurtent à une explosion combinatoire. De très nombreux algorithmes ont été proposés pour réaliser des extractions complètes de tous les ensembles fréquents dans des contextes réalistes. Pour notre système, nous avons utilisé une implémentation de *FP-Tree* [51] étendue à la gestion des motifs fermés [88]. RE-Hackle n'est pas pertinent dans ce contexte sans conrainte par expression réguliere. La structure FP-Tree a fait ses preuves lors des extractions de très grande échelle. Elle est adaptée aux tâches courantes en finance quantitative qui ne concernent généralement que quelques milliers d'ensembles. Notre publication [8] contient un exemple pas-à-pas facile à suivre pour comprendre l'extraction des motifs fréquents avec la structure FP-tree. L'extraction des signatures de bourse se fait quasiment en temps réel, ce qui rend la méthode proposée éligible au trading haute fréquence.

5.2 Le système de trading

Le comportement du sous-jacent est assujetti à des phénomènes de mode et à des régimes de trading qui varient dans le temps. Aussi, est-il nécessaire de suivre la va-

lidité des signatures dans le temps, d'enrichir la base des signatures par les nouveaux motifs qui apparaissent dans les cours et d'écarter (invalider) les motifs qui commencent à sous-pérformer[1] par rapport à l'indicateur seul. Nous décrivons maintenant ce mécanisme d'apprentissage progressif. Les dates T_0 à T_n déterminent $n-1$ périodes. La première est utilisée pour l'initialisation du système (apprentissage des bonnes signatures) qui seront utilisées pour le trading pendant la période suivante. A partir de l'apprentissage initial, chaque période sert à traiter avec les bonnes signatures identifiées jusqu'au début de la période et alimente les signatures de la période suivante.

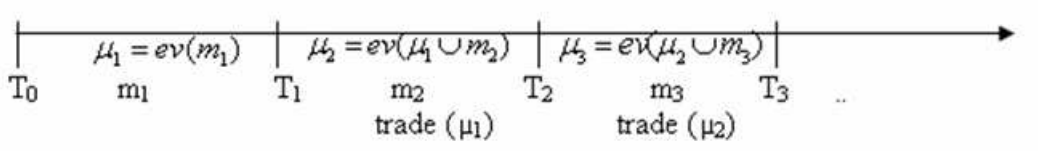


FIG. 5.4 – *Découpage en périodes.*

La première période fournit les motifs m_1. Cet ensemble est comparé à la performance de l'indicateur seul pendant cette même période pour n'en garder que les motifs μ_1 qui améliorent les performances de l'indicateur. Le contenu de μ_1 générera des transactions de la deuxième période, et sera ajouté aux motifs m_2 de cette période pour calculer l'ensemble μ_2, contenant les signatures de bonne qualité identifiées jusqu'à T_2 (la fin de la deuxième période). Le système de trading bâti autour de ce re-balancement est illustré sur la Figure 5.5. Suivons d'abord les flèches qui émergent de la période $p-1$ jusqu'au rectangle filtrage. Cette partie correspond à l'apprentissage du système. L'étape d'extraction fournit les motifs fréquents de la période $p-1$. La mesure de performance évalue les gains de chaque motif fréquent identifié. Pour ce faire, la stratégie de trading à optimiser est effectuée sur la période $p-1$ et l'on calcule la performance de l'indicateur seul. Ensuite, pour chaque motif fréquent extrait, la stratégie est à nouveau appliquée. Cette fois, l'indicateur n'est retourné vers la stratégie que si son scénario d'occurrence contient le motif en cours d'étude. Si la performance de la stratégie précédée par un motif est supérieure à la performance de l'indicateur seul, le motif est ajouté à l'ensemble μ_{p-1}, i.e., la collection des bonnes signatures, et utilisé par le trading de la période suivante. Les indicateurs de la période p sont pris en compte uniquement si leur scénario d'occurrence contient une bonne signature de la période précédente. Dans ce cas, le filtrage génère un signal qui est envoyé à la stratégie. Après le trading, la période p devient une période d'apprentissage pour la période $p+1$.

Un système de trading sur trois périodes (apprentissage, sélection, trading) est difficile à mettre en œuvre sans une diminution drastique de la fréquence des transactions. En fait, les signatures sont attachées à un régime de marché et leur apport disparaît si elles ne sont pas appliquées immédiatement en trading (impossibilité d'avoir une période de sélection). Le raccourcissement de la période d'apprentissage est délicat, car la diminution de points pour l'étape de fouille de données conduit à un sur-apprentissage, i.e., un phénomène de sur-spécialisation qui produit des bonnes performances pendant la période d'apprentissage

[1]Le jargon financier utilise les expressions sur-pérformer et sous-pérformer pour comparer les performances des stratégies financiers.

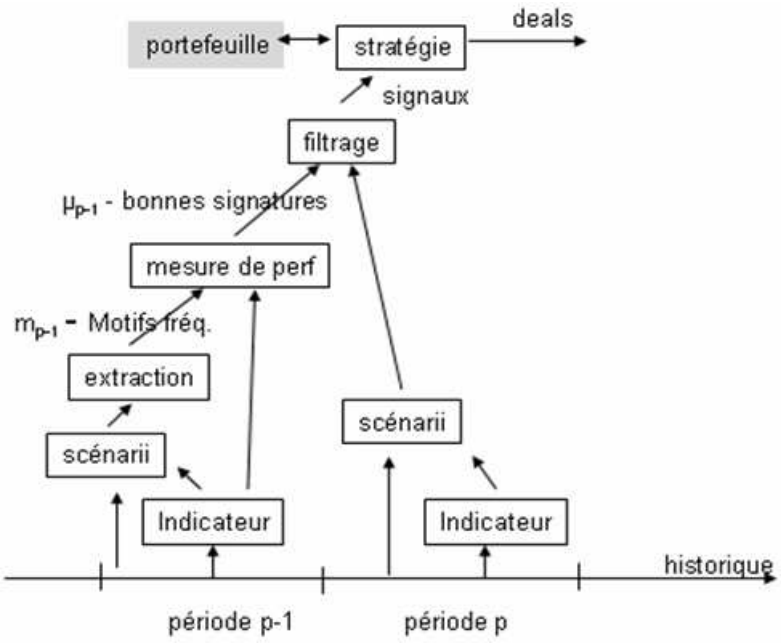


FIG. 5.5 – *Système de trading.*

mais qui se comporte de manière médiocre ultérieurement.

5.3 Résultats expérimentaux

Afin de démontrer le caractère générique de la méthode proposée, nous l'avons appliqué à des données financières de type *equity* (actions) et *fixed income* (taux d'intérêt), qui ont des comportements (volatilité, retour à la moyenne, etc.) très différents. Les indicateurs utilisés sont simples et bien connus dans le monde financier : Williams %R, croisement de moyennes mobiles, ainsi qu'un indicateur confidentiel, appelé boite noire (BN). La période étudiée s'étend sur environ cinq années. La performance de chaque transaction a été mesurée en terme de gain réalisé 5 jours après son origine. Le Williams %R permet de déterminer quand un actif est sur vendu ou sur acheté sur une période (d'habitude 2 semaines).

$$W\%R = -100\frac{\max(H_i) - C}{\max(H_i) - \min(L_i)}, \text{ où i =7 } \ldots 15 \text{ jours} \tag{5.1}$$

H_i et L_i représentent les plus haut (resp plus bas) journaliers du cours d'un actif, C est le dernier cours de clôture avant de calculer l'indicateur.

Le portefeuille est composé d'un dépôt *cash* et du nombre de titres détenus. Dans la gestion, la valeur du portefeuille, *wealth process*, ne tient pas compte de la rémunération du dépôt *cash* (ni de l'effet des dividendes pour les données de type *indice* ou *equity*, car l'étude vise à montrer l'efficacité du filtrage toutes choses égales par ailleurs. Les tests concernent uniquement des stratégies de type *long*, sans vente à découvert.

5.3.1 Données type actions

Le Tableau 5.1 compare les performances de W%R seul et de W%R signé sur une action de l'indice CAC40. Chaque ligne du tableau correspond à un an. Les signatures réduisent le nombre de mauvaises transactions et surpassent l'indicateur W%R non-sigé.

	W%R signé		W%R non-signé	
Année	Perf	nbTrades	Perf	nbTrades
2	101.74	88	99.81	144
3	107.06	142	107.14	174
4	100	0	102.06	190
5	98.3	80	97.25	96
6	104.57	134	104.05	150
7	98.55	108	99.44	118
8	112.58	166	113.26	184
9	146.95	180	146.92	196
10	88.62	70	88.68	130
11	96.84	104	92.41	112
12	80.45	72	78.68	180
13	89.37	90	92.97	100
14	99.16	130	96.67	144
15	95.38	38	95.58	48

TAB. 5.1 – *Le W%R signé sur performe le W%R seul sur 15 ans. L'ordre de grandeur des transactions restant après filtrage est comparable au nombre de transactions au départ.*

L'indicateur signé ne surpasse pas l'indicateur seul sur toutes les périodes. Elle a néanmoins un meilleur comportement en moyenne. L'analyse des actions composant l'indice CAC40 (cf. Annexe B) depuis le 1^{er} janvier 2000 a permis de valider l'apport de la technique des signatures sur un grand nombre de périodes et d'actions (200 points). Sur 5 ans, la méthode proposée a apporté 6% de gains en plus par rapport au W%R non signé. Compte tenu du nombre de périodes comparées, le test t de Student (cf. tableau 5.2) à permis de conclure que les performances de la technique signé sont supérieurs à l'indicateur seul à un seuil de confiance de 10%. Pour une stratégie quantitative dans la finance ce gain est significatif !

Le lecteur peut se rapporter à l'Annexe B pour consulter les données de tests. Nous avons également testé l'apport des signatures sur un indicateur propre à l'entreprise CIC et qui est appelé boîte noire (BN). Comme le tableau 5.3 l'illustre, l'amélioration est encore plus significative, et atteint le seuil de 1%.

	W%R signé	W%R seul
Moyenne	96.93	90.63
Variance	5264	13906
Observations	200	
Corrélation	0.82	
Différence hypothétique des moyennes	199	
Statistique t	1.25	
P (T¡t) unilatéral	0.11	
Valeur critique de t	1.65	

TAB. 5.2 – *L'apport des signatures est significatif pour l'indicateur Williams%R. Meme stratégie de trading pour les deux cas.*

	BN signée	BN seule
Moyenne	104.22	98.13
Variance	2132	3940
Observations	200	
Corrélation	0.81	
Différence hypothétique des moyennes	199	
Statistique t	2.31	
P (T¡t) unilatéral	0.01	
Valeur critique de t	1.65	

TAB. 5.3 – *L'apport des signatures à l'equity est significatif pour l'indicateur Boite Noire.*

5.3.2 Données de type "taux d'intérêt"

Compte tenu de l'efficience accrue[2] des marchés de capitaux, le W%R ne dégage pas de performance sur la période étudiée. Cependant, les signatures permettent de baisser les pertes. Comme les graphiques suivants le montreront, par période le W%R signé produit des gains alors que le W%R s'inscrit dans une tendance baissière. La figure suivante illustre le *wealth process* de la stratégie, i.e., la valeur du portefeuille produit par la stratégie tout au cours de sa vie.

La table 5.4 illustre les arguments développés pour la figure 5.6. Les chiffres ont été obtenus avec remise des capitaux, i.e., le portefeuille est vidé en début de chaque période et le dépôt *cash* est réinitialisé à 100.

5.3.3 Discussion

La stratégie de trading en réponse de l'indicateur a été paramétrée pour mesurer les performances à N jours après la transaction (5 jours pour les statistiques précédentes). Les signatures caractérisent une configuration de marché ponctuelle et ont un pouvoir

[2]Ce marché est dominé par les investisseurs institutionnels qui ne génèrent pas les mêmes motifs que les particuliers intervenant sur les actions. Dans certaine mesure les indicateurs techniques proviennent d'un phénomène auto-entretenu : la réaction de masse à un motif donné déclenche des transactions qui décalent le marche comme prédit par l'indicateur.

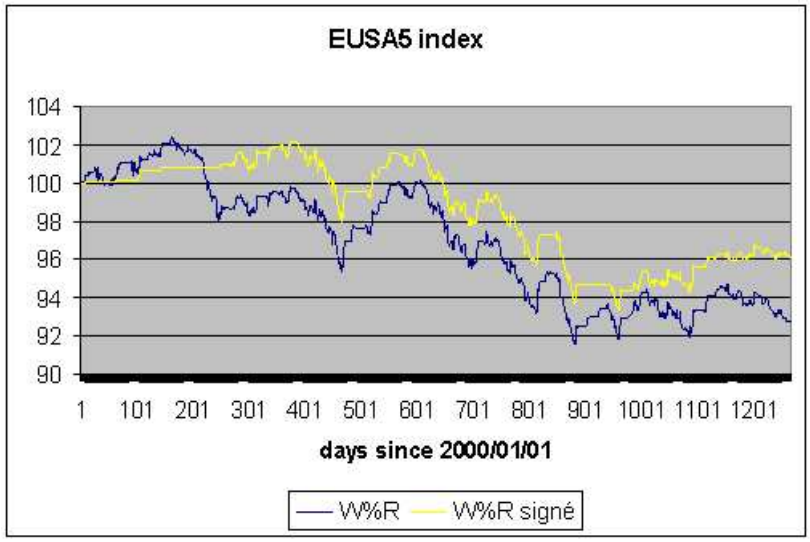


FIG. 5.6 – *Le W%R signé évite souvent les périodes où le W%R seul perd, voir il réalise des gains (de 92.93 à 96.13, soit 3.2% de mieux)*

	W%R signé		W%R seul	
Année	Perf	nbTrades	Perf	nbTrades
2000	100.97	24	98.62	356
2001	98.35	306	98.73	334
2002	98.83	346	98.40	382
2003	96.81	266	98.14	310
2004	101.185	268	98.83	356

TAB. 5.4 – *Comparaison des performances de W%R signé et W%R nu appliqué sur les taux swap 5 ans entre début 2000 et fin 2004. (1999 utilisée pour initialiser l'apprentissage)*

de prédiction d'environ une semaine. Il est normal que leur validité soit limitée dans le temps : l'ensemble qui décrit la configuration de marché n'inclut aucune information sur le comportement historique du cours. La *mémoire longue* et l'*auto corrélation* ne sont pas utilisées. Nous pensons qu'en enrichissant la description des scénarii par des éléments du passé (minimum et maximum sur un passé plus long, moyenne mobile sur Y jours) le pouvoir de prédiction peut être étendu de quelques jours, mais l'extraction des motifs fréquents devient plus difficile à cause d'un alphabet plus grand. La durée de vie d'une signature est assez courte, et dépasse rarement trois périodes (y compris celle d'apprentissage). Dans le cas du W%R, les signature surpassent la stratégie nue en perdant un peu moins d'argent. On doit alors se demander si l'amélioration des performances n'est pas essentiellement liée au simple filtrage ? En effet, appliquer moins souvent une stratégie perdante améliore forcément les performances. Cependant, dans le cas d'un filtrage aléatoire, l'écart entre les deux courbes devrait se constituer progressivement, alors que sur la figure 5.6 l'indicateur filtré se distingue aux moments ou l'indicateur seul commence à perdre. De plus, le filtrage produit des résultats significativement meilleurs pour l'indicateur BN, et présente

des gains alors que la stratégie de base est légèrement déficitaire, cf. Tableau 5.3.

Stratégies croisement des moyennes mobiles Ce type d'indicateur est basé sur la mémoire longue non corrélée avec les signatures qui caractérisent la configuration présente. Aussi, nous avons constate lors des expérimentations que l'amélioration de ce type d'indicateur par la méthode des signatures ne produisait aucune amélioration significative.

Frais de transactions L'effet du filtrage par les signatures est de limiter le nombre des transactions en gardant seulement celles qui ont le plus de chances d'aboutir. En conséquence, il réduit également les frais de transactions. L'étude présente sous-estime donc l'apport des signatures d'une part en négligeant les économies sur les frais de transactions et d'autre part en ignorant la rémunération du compte de dépôt.

5.4 Conclusion

Les tests effectués promettent d'aboutir à un système de trading automatique. Le "back testing" des indicateurs avec re-balancement s'est montré capable d'identifier les signatures caractéristiques à chaque période. Elles sont suffisamment stables dans le temps pour pouvoir améliorer les performances de l'indicateur technique durant la période suivante. Les techniques décrites apportent donc un moyen pour améliorer les stratégies quantitatives existantes et ne demandent pas la mise en œuvre de nouvelles stratégies. Dans les exemples traités, la technique des signatures a produit entre 3% à 6% de gains en plus par rapport à l'indicateur seul. Nous avons considéré trois indicateurs : le W%R et le croisement de moyennes mobiles et un indicateur confidentiel à l'entreprise CIC.

Sur le plan algorithmique, nous avons suivi une démarche originale. Les données de départ sont un vecteur à cinq éléments sur quelques jours, donc une série temporelle multi-dimensionnelle. La représentation ensembliste présentée dans ce chapitre permet de transformer des données séquentielles multi-dimensionnelles dans des ensembles simples bien connus afin de les traiter avec des méthodes des ensembles fréquents.

Cette application a utilisé des ensembles fermés afin de caractériser les signatures des scénarii de marché. Ce choix a été dicté par la nécessité de réduire le nombre des motifs retournés, et de s'assurer que l'intensité de chaque signature est identique : en effet, si a et abc sont des motifs fermés, le support de ab sera au moins égale au support de abc. Nous pouvons ainsi éviter d'exagérer l'importance de certaines signatures, car l'identification de ab et abc dans le même scénario produirait un signal deux fois plus puissant que abc seul. Ceci est d'autant plus important, que les stratégies de "trading" basées sur une conjonctions d'indicateurs techniques cherchent à identifier les situations oòu plusieurs indicateurs techniques indiquent la même action (achat ou vente), produisant un décalage plus important du marché. Les motifs fermés permettent de différencier les situations quand deux signatures différentes donnent la même prédiction des cas ou un motif maximal serait amplifié par l'un de ses sous-motifs. La portée des motifs fermés va au-delà d'une simple représentation condensée visant à réduire le coût de stockage : ils permettent ici de faire émerger une caractérisation du phénomène sous-jacent aux données.

Chapitre 6

Valorisation des Produits Dérivés

6.1 Introduction à la Valorisation par Monte Carlo

Soit $T_0, T_1, \ldots, T_N < T$ une série de dates successives, et Y_t, un processus stochastique qui détermine l'évolution des paramètres de marché, i.e., les cours des actifs financiers. En général, le processus Y_t est un mouvement Brownien multi-dimensionnel, mais d'autres choix plus complexes sont également possibles. Soit $(\Omega, \mathcal{F}, \mathbb{P})$ un espace de probabilités, où $\mathcal{F} = \{\mathcal{F}_t : 0 \leq t \leq T\}$ est la filtration générée par le processus $Y(\omega, t)$ (noté couramment Y_t).

Un *produit financier* est un contrat qui définit les flux d'argent (*cash flow*) $C_{T(1)}$, $C_{T(2)}, \ldots, C_{T(n)}$ échangés entre deux parties à une succession de dates $T_1, T_2, \ldots, T_n$. Ces flux dépendent des futures réalisations des données de marché (références). Un tel contrat est appelé *option* dans le cas particulier où le choix de l'une des parties peut modifier les cash flows pendant la durée de vie du produit : renoncer à la réception de certains cash flows, annuler le contrat, changer la référence, etc. Sans doute l'exemple classique est l'*option call vanille* de strike K sur le sous-jacent S définie par :

$$C_T(S_T) = \max(S_T - K, 0) \tag{6.1}$$

L'acheteur de cette option a le droit de recevoir la différence entre le niveau du sous-jacent S à la date T et un niveau prédéterminé K, appelé *strike*. Il exercera son droit seulement si cette valeur est positive.

Le théorème fondamental de la finance affirme que la valeur d'un tel contrat est égale à l'espérance[1] actualisée des futures cash flow. En considérant un seul *cash flow* en T :

$$V_t = N_t E^{\mathbb{N}}\left(\frac{C_T(S_T)}{N_T} \mid \mathcal{F}_t\right) \tag{6.2}$$

Cette équation est formellement ré-introduite dans la section 6.3.1. On se contentera de dire que C_T représente un seul cash flow à la date T, N_t est le numéraire qui normalise les cash flows perçues, V_t la valeur actuelle, en t. L'espérance est prise sous la probabilité équivalente $\mathbb{N}$ sous laquelle N_t est une martingale.

[1] L'espérance est prise dans une mesure forward neutre sous un numéraire N_t (voir [21] pour une introduction ou [45] pour un traitement rigoureux de l'usage des numéraires dans le pricing des produits dérivés).

Une possibilité pour calculer l'espérance dans l'equation (6.2) est d'intégrer analytiquement C_T/N_T sous la probabilité $\mathbb{N}$. Cette approche directe est peu efficace en dimension élevée [95], et il est en général préférable de réaliser une intégration numérique par la méthode Monte Carlo, dès que le cash flow dépend de plus de trois sources d'incertitude.

La valorisation par la méthode de Monte Carlo est basée sur une analogie entre probabilité et volume. Selon la théorie de la mesure, la probabilité d'un événement est égale à la mesure (volume) de l'événement divisée par la mesure de l'univers de probabilités. La méthode de Monte Carlo emprunte le chemin inverse : le volume correspondant à un ensemble (événement) est obtenu en calculant la probabilité de l'événement (ensemble). Dans la pratique, l'univers de probabilités est rempli de points tirés aléatoirement, et le nombre de points recueillis par l'ensemble est interprété comme son volume. Ce volume, divisé par le volume de l'univers correspond à la probabilité de l'événement en question. La loi des grands nombres assure que l'estimation Monte Carlo converge vers la vraie valeur et le théorème de la limite centrale donne la vitesse de convergence, qui est proportionnelle à l'inverse de la racine quarrée du nombre de points utilisés.

Le volume est facilement relié à la valeur d'une intégrale. En supposant une distribution uniforme de y, l'intégrale I de $f(y)$ sur le volume V

$$I = \int_V f(y)dV = E\left(f(y)\right) \tag{6.3}$$

est estimée comme

$$\widehat{I} = E\left(f(y)\right) = \frac{\sum^{MC} f(y_i)}{MC} \text{ , où } MC \text{ est le nombre de tirages Monte Carlo.} \tag{6.4}$$

La méthode Monte Carlo génère des scénarii d'évolution des paramètres de marché, appelé *chemins*. En employant un langage probabiliste, chaque chemin Monte Carlo est un *trajectoire* échantillonné aux dates $T_1, T_2 \ldots, T_n$ du processus Y_t, i.e., une suite discrète des réalisations $y_i(T_1), y_i(T_2), \ldots, y_i(T_n)$.

6.1.1 Options bermuda

Un produit rappelable P^r de type bermuda[2] est un produit financier optionnel que l'une des parties peut arrêter (moyennant éventuellement le payement d'une pénalité) à une série de dates prédéfinies $T_1, T_2, \ldots, T_n$. Ce mémoire se focalise sur l'étude des rappels sans paiement de pénalité. Un produit rappelable peut être représenté comme la somme du même produit sans possibilité de rappel P, plus une option de bermuda (de rappel) qui donne droit à mettre en place une série de paiements démarrant en $T_i \in T_1, T_2 \ldots T_n$ et neutralisant les paiements du produit P.

La valorisation d'une option bermuda est un problème de contrôle stochastique complexe (voir [58] ou [53] pour une introduction). Supposant un processus markovien, l'équation de Hamilton-Jacobi-Bellman pour la valeur d'une option bermuda (de rappel) se réduit à l'expression suivante :

[2] A distinguer du type américain, qui peut être arrêté à tout moment.

$$V_{T(i)} = \begin{cases} \max\left(0, C_{T(i)} + N_{T(i)} E\left(\frac{V_{T(i+1)}}{N_{T(i+1)}} \mid \mathcal{F}_{T(i)}\right)\right) \text{ , si } i < n \\ \max\left(0, C_{T(n)}\right) \text{ , si } i = n \end{cases} \tag{6.5}$$

qui peut être évaluée par rétro-propagation (*backward induction*). Le premier terme dans la fonction *max* correspond à la valeur des cash flow si l'option est exercée en T_i, le deuxième terme est appelé valeur de continuation.

6.1.2 Markovisation des processus

L'état du marché sera décrit par le vecteur Y_t, qui contient les cours des sous-jacents. à tout moment T_i. L'expression du cash flow $C_{T(i)}$ du produit financier est une fonction h de l'évolution du marché entre t et T_i.

$$C_{T(i)} = h(Y_{t(1)}, Y_{t(2)}, \ldots), \text{ où } t_1, t_2 \ldots \leq T_i \tag{6.6}$$

La fonction h est appelé payoff. Un bon exemple est donné par l'equation (6.1), avec $h(x) = \max(x - K, 0)$, mais le payoff est potentiellement une fonction de dimension infinie. Le cas le plus courant correspond à la situation particulière quand le cash flow $C_{T(i)}$ dépend seulement du dernier état du marché $Y_{T(i)}$. Tous les autres cas (produits) sont appelés *path dependent*.

Le théorème suivant définit la famille de payoffs (produits) que nous étudions ici.

Théorème 3. *Si la valeur $V(T_n)$ du produit financier vu depuis T_n dépend seulement d'un nombre d'observations fini $Y_{o1}, Y_{o2}, ..Y_{om} \in \mathcal{F}_{T_N}$, au moments $T_{o1}, T_{o2}, ..T_{om}$, et Y_t est au plus discrètement path dependent* [3]*, i.e. $\forall s \leq t$,*

$$E(Y_t \mid \mathcal{F}_s) = E(Y_t \mid X_s) \tag{6.7}$$

avec X_s défini comme la source d'incertitude augmentée par

$$X_s = Y_s \cup \{Y_{oi}, T_{oi} \leq s\} \tag{6.8}$$

on peut alors écrire sans perte d'information que

$$E^{\mathbb{N}}(V_N \mid \mathcal{F}_t) = E^{\mathbb{N}}(V_N \mid X_t) \tag{6.9}$$

Démonstration. Immédiat, par application des propriétés de l'espérance conditionnelle et le fait que Y_{oi} dépend seulement de $\{Y_{o1}, Y_{o2}, ..Y_{om} | T_{om} < T_{oi}\}$.

$$\begin{aligned} E^{\mathbb{N}}(V_N \mid \mathcal{F}_t) &= E^{\mathbb{N}}(V_N \mid X_t \cup \mathcal{F}_t - \sigma(X_t)) \\ &= E^{\mathbb{N}}(V_N \mid X_t) \end{aligned}$$

□

[3] Aucune contrainte posée sur les dérivés du prix, les greeks.

Le changement de variable proposé par le théorème 3 remplace la variable non Markovienne Y_t en une nouvelle variable augmentée X_t Markovienne.

Remarques. L'augmentation du vecteur de *l'état du monde* est possible seulement si le nombre d'observations est fini. Théoriquement, tout instrument discret (y compris les corridors) pourrait utiliser la technique de l'augmentation, même si les observations journalières résulteraient en un processus X_t de dimension trop élevée pour une implémentation efficace. Les observations continues comme dans le cas des options à barrière[4] ou des options lookback[5] ne pourront pas être markovisées avec les techniques proposées dans ce mémoire et sont donc exclues de notre étude. Fort heureusement, la majorité des options négociées est définie sur un nombre d'observation réduits, quelques dizaines au plus, ce qui donnera une variable d'état markovienne d'assez faible dimension, utilisable pour la simulation numérique.

6.2 État de l'art des Monte Carlo américains

L'intégration Monte Carlo est la méthode de valorisation dominante des produits "'path dependent"' complexes. Malheureusement, elle est peu adaptée au calcul des prix en utilisant le rétro-propagation définie dans equation (6.5). Revoyons trois techniques pour étendre le champs d'application de la méthode Monte Carlo à la valorisation des options bermuda.

- **Barraquand et Martineau** construisent un maillage spatio-temporel et calculent les probabilités de transitions entre les cellules par diffusion Monte Carlo. Ensuite, le maillage est évalué par rétro-propagation. Cette approche utilise donc la programmation dynamique pour approcher la valeur des options bermuda, en évaluant directement equation (6.5).
- **Longstaff et Schwartz** effectuent une régression de la valeur de continuation sur une famille de fonctions des variables d'état (composantes de X_t), et évaluent ensuite equation (6.5) sur les valeurs regressées.
- **Andersen** cherche à paramétrer directement la frontière d'exercice[6]. Sa méthode comporte deux phases : d'abord il optimise les seuils H_i, tel que la stratégie d'exercice définie par le *temps d'arrêt* $X_{T(i)} < H_i$ ait valeur maximale sur une population de chemins Monte Carlo d'apprentissage. Ensuite, il applique les seuils trouvés sur une nouvelle population et obtient la valeur du produit.

[4]Cash flow payé seulement si X_t n'a pas franchi un seul prédéterminé pendant la durée de vie de l'option, i.e., $h(X_i) = 1_{X(s)<b;\forall t<s<T(i)}$.

[5]Cash flow égal à la différence du maximum de X_t réalisé pendant la durée de vie de l'option moins le strike de l'option. $h(X_i) = \max\left(\sup_{t<s<T(i)} X_s - K, 0\right)$.

[6]Suite des variables de décision indexée par le temps, qui sépare la région où une option bermuda est exercée de celle ou elle est laissée continuer. Voir Lamberton et Lapeyre [58], chapitre 2 pour une définition formelle avec l'enveloppe de Snell.

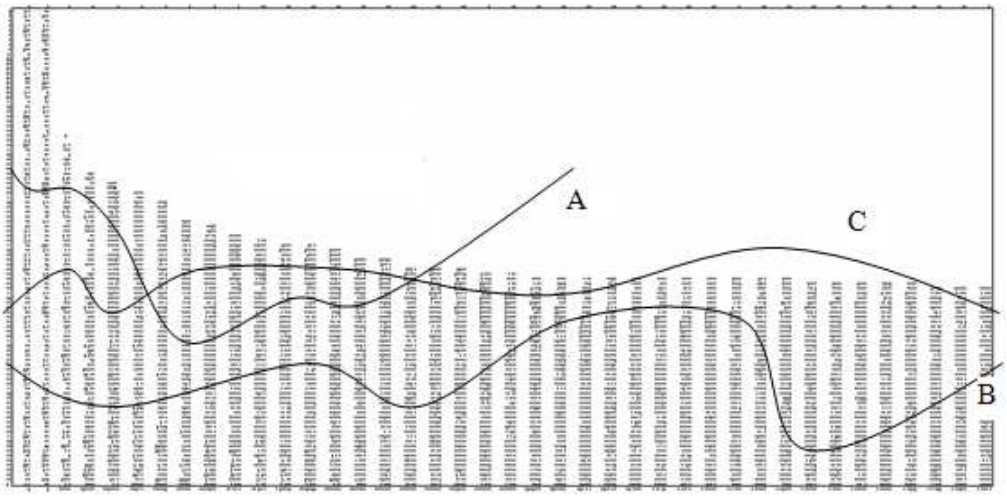


FIG. 6.1 – *La projection de la frontière d'exercice dans le plan (valeur, temps) affiche un comportement très régulier. Les chemins A, B et C représentent 3 réalisations Monte Carlo.*

6.2.1 Maillage spatio-temporel de Barraquand-Martineau

La méthode de Barraquand-Martineau considère qu'à chaque date de décision (de rappel) $T_i \in T_1 \dots T_n$, la variable de décision (d'état) x_i est unidimensionnelle. La méthode calcule les bornes B_j^i, $j \in 1 \dots bins$ ($bins$ est un paramètre fixe par l'utilisateur, en général 50-100) telle que chaque intervalle $\left[B_j^i \dots B_{j+1}^i\right)$ contienne à peu près le même nombre de chemins Monte Carlo, pour lesquels $x_i \in \left[B_j^i \dots B_{j+1}^i\right)$. La figure 6.2 illustre ce principe. Les bornes B_j^i délimitent des partitions de l'espace de valeur de la variable d'état x_i. Chaque partition (bin) de l'espace contiendra (a) la valeur normalisée par $N_{T(i)}$ des cash flows en T_i ($\tilde{V}_{i,j}$), et (b) les probabilités de transition depuis B_j^i vers les bins suivants B_k^{i+1}. Cette probabilité de transition $\pi_{i,j,k}$ est calculée de façon empirique, en divisant le nombre de trajectoires allant de B_j^i vers B_k^{i+1} par le nombre de trajectoires entrants en B_j^i.

$$\pi_{i,j,k} = \frac{Card\left\{\sharp m \mid x_i^m \in \left[B_j^i \dots B_{j+1}^i\right) \ \&\ x_{i+1}^m \in \left[B_k^{i+1} \dots B_{k+1}^{i+1}\right)\right\}}{\sum_{k=1..bins} Card\left\{\sharp m \mid x_i^m \in \left[B_j^i \dots B_{j+1}^i\right)\right\}} \tag{6.10}$$

En définissant l'espérance interne de l'equation (6.5) comme

$$E\left(\frac{V_{T(i+1)}}{N_{T(i+1)}} \mid \mathcal{F}_{T(i)}\right) = \sum_{k=1\dots bins} \pi_{i,j,k} \tilde{V}_{i+1,k} \tag{6.11}$$

il est intéressant à noter que x_i est $\mathcal{F}_{T(i)}$ mesurable, et la connaissance de $\mathcal{F}_{T(i)}$ donnera aussi l'indice j de $\pi_{i,j,k}$, puisqu'il est facile de trouver les bornes B_j^i, B_{j+1}^i contenant x_i. L'equation (6.11) fournit une valeur pour chaque partition (bin) du maillage. En

procédant par rétro-propagation, il est possible de calculer la valeur de l'option bermuda. Le lecteur attentif aura remarqué que nous calculons ici des espérances conditionnelles $\hat{V}_{n,k} = E\left(C_{T(n)} \mid B_k^n \leq x_n < B_{k+1}^n\}\right)$.

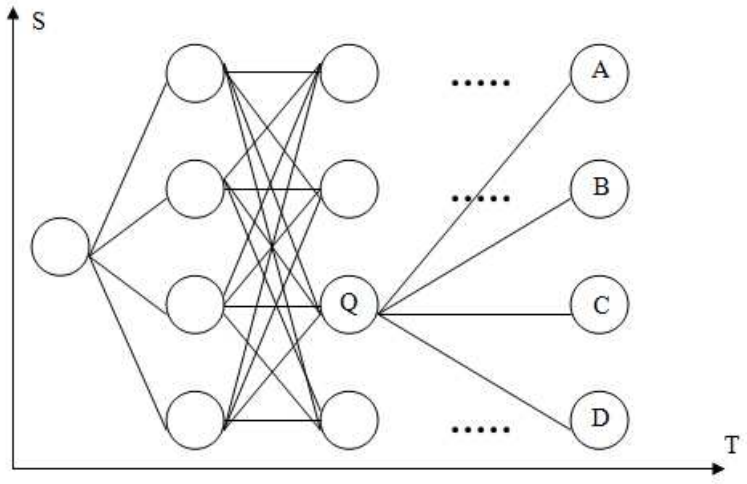


FIG. 6.2 – *Le partitionnement temps-espace Markovien de Barraquand et Martineau.*

6.2.2 Régression multilinéaire : LSM

Longstaff et Schwartz [63] emploient une autre estimation de l'espérance conditionnelle interne de l'equation (6.5). A chaque date d'exercice T_i, ces auteurs permettent que la variable d'état x_i soit un vecteur à plusieurs éléments, et définissent une famille de fonctions de base f_j^i, pour que

$$E\left(\frac{V_{T(i+1)}}{N_{T(i+1)}} \mid \mathcal{F}_{T(i)}\right) = \sum_j f_j^i(x_i) \tag{6.12}$$

Dans cette méthode, la frontière d'exercice sera paramétrée par les coefficients de la famille f_j^i, qui peut être une collection de polynômes ou de fonctions exponentielles. La méthode de Longstaff et Schwartz opère en deux phases : d'abord il génère une première collection des chemins Monte Carlo pour déterminer les coefficients de la famille f_j^i par régression multilinéaire, et ensuite une seconde collection de chemins Monte Carlo sert à calculer le prix de l'option bermuda.

6.2.3 Paramétrage de la frontière d'exercice :la méthode d'Andersen

La méthode d'Andersen [12] s'appuie sur la notion du *temps d'arrêt*[7]. La valeur de l'option bermuda est aussi exprimée avec l'équation.

$$V_t = N_t \sup_{\tau} E\left(\sum_{s\geq\tau} \frac{h(x_s)}{N_s} \mid \mathcal{F}_t\right) \tag{6.13}$$

où τ est un temps d'arrêt, i.e., une stratégie de décision qui dira s'il faut exercer l'option bermuda ou continuer d'attendre un moment plus favorable. Parmi tous les temps d'arrêts possibles, Andersen choisit une stratégie simple

$$\widehat{\tau} = \min\left(i \in \{1, 2, \dots n\} : x_i \geq H_i\right) \tag{6.14}$$

La suite $H_i \in \mathbb{R}$ exprime les valeurs d'un paramètre de marché x_i à chaque date d'exercice T_i, correspondant à la valeur de la frontière d'exercice uni-dimensionnelle. Si le paramètre x_i est supérieur H_i il y a exercice, sinon on continue d'attendre. Évidemment, le temps arrêt d'Andersen est calculé sur un sous-ensemble de tous les temps d'arrêts possibles, et son estimation est inférieure à la valeur réelle.

$$\widehat{V_t} = N_t E\left(\sum_{s\geq\widehat{\tau}} \frac{h(x_s)}{N_s} \mid \mathcal{F}_t\right) \leq V_t \tag{6.15}$$

Contribution. Dans la pratique, pour les produits les plus courants, l'estimateur de equation (6.15) est assez bonne. Plus le paramètre x_i est proche du prix du produit en T_i, plus l'estimation d'Andersen se rapprochera de la valeur exacte. Utiliser le prix exact produirait des résultats parfaits, mais cette approche est extrêmement couteuse en temps de calcul. Notre *contribution originale* est de proposer une approximation pour le prix du produit facile à calculer par le biais d'une technique baptisée *neighbourhood Monte Carlo*.

6.3 Extraction de la valeur résiduelle conditionnelle

La variable de décision dans la méthode d'Andersen ou le prix d'exercice de Longstaff-Schwartz nécessitent de calculer la valeur des futurs cash flows en fonction de l'état du marché au moment de la décision d'exercice/continuation. Quand il n'y a pas de formule fermée pour la valeur conditionnelle, les professionnels cherchent a approximer la valeur conditionnelle, ce qui conduit à une décision sub-optimale, ou lancent une deuxième simulation Monte Carlo pour calculer l'espérance exacte, ce qui est gourmand en temps de calcul, et donc souvent irréaliste.

Nos travaux apportent une solution à ce problème. L'une des principales contributions de cette thèse est de réutiliser les chemins de la simulation Monte Carlo primaire pour inférer l'espérance du deuxième Monte Carlo conditionnel [9]. Ce chapitre prouve que l'espérance calculée avec la nouvelle méthode converge vers la valeur correcte.

[7] Le temps d'arrêt est une variable aléatoire $\tau : \Omega \to \mathbb{R}$ telle que $\{\tau \leq t\} \in \mathcal{F}_t$ (voir chapitre 2 de [58] pour une application à la finance).

La littérature des simulations Monte Carlo connaît quelques travaux qui cherchent à estimer la valeur des futurs cash flows conditionnels à partir de chemins Monte Carlo [27]. Il existe également des implémentations double Monte Carlo dans les salles des marchés mais celles-ci ne sont évidemment pas publiées. La vitesse de calcul est une composante essentielle de tout système de production réussi. A notre connaissance, aucune tentative n'a connu un succès en dehors des départements de recherches des banques d'investissement. Les modèles actuels évitent de calculer l'espérance conditionnelle coûteuse, et se contentent d'utiliser des hypothèses simplificatrices comme la caractérisation uni-dimensionnelle de la méthode d'Andersen, ou les valeurs approximatives de Longstaff et Schwartz.

Toujours à notre connaissance, personne n'a essayé de combiner les simulations Monte Carlo avec la recherche de k plus proches voisins. Dernièrement, ce domaine a reçu beaucoup d'attention, et ses récentes avancées peuvent apporter un vrai plus à la valorisation des produits dérivés. Nous avons appliqué avec succès la recherche de k plus proches voisins pour calculer de façon très efficace la valeur conditionnelle à un chemin Monte Carlo.

Soit X_t une variable d'état augmentée comme dans equation (6.8) contenant toute l'information caractérisant l'état du monde au moment t, i.e., les fixings en t, et un nombre fini d'observation historiques sur le chemin courant. Notre *idée clé* est de remplacer l'espérance conditionnelle exacte par une autre espérance conditionnelle moins contraignante mais plus facile à calculer, qui converge vers l'espérance conditionnelle exacte au niveau de précision souhaité, i.e., au lieu de conditionner exactement par l'état x, notre méthode utilisera la condition être dans le ϵ-voisinage de x. L'état a été défini par equation (6.8).

$$E(V_n \mid X_t = x) \approx E(V_n \mid X_t \in (x - \epsilon, x + \epsilon)) \tag{6.16}$$

La puissance de l'espérance conditionnelle affaiblie de equation (6.16), c'est qu'elle peut être calculée sans démarrer une seconde simulation Monte Carlo depuis l'état x. La recherche des k plus proches voisins de x avec la méthode proposée par Arya et al. [13, 14] permet de recueillir suffisamment d'information pour estimer cette espérance conditionnelle au niveau de précision souhaité.

La méthode proposée, baptisée neighbourhood Monte Carlo (nMC), fait donc deux hypothèses : (a) elle projette un payoff path dépendant dans un processus Markovien, et (b) elle approxime l'espérance conditionnelle exacte par un condionnement moins contraignant.

Neighbourhood Monte Carlo est théoriquement justifié quand la projection de payoff path dépendant dans un processus Markovien est possible, i.e., quand la valeur est en plus discrètement path-dependent. En général, ceci ne réduit pas beaucoup le champ d'application de la méthode, car les payoffs usuels dépendent rarement de l'historique complet. D'habitude, la dépendance des payoffs de l'historique se limite à quelques quantités comme le maximum ou minimum discrètement observé des cours sous-jacents, ou un incrément qui s'accumule période après période à la snowball[8]. Dans ce cas, il suffit de connaître seulement la valeur du coupon précédent et davantage d'information historique sur la genèse du coupon est inutile pour calculer le coupon en cours. La seconde hypothèse paraît plus forte, car la condition est rendue plus incertaine. La section 6.3.2 prouve que la nouvelle

[8]Produit dérivé fortement path-dependent sur le marché des taux ou matières premières, qui paye $\max(\text{coupon précédent} + \text{coeff} \times (\text{Sous-jacent} - \text{Strike}), 0)$. Il est souvent rappelable par l'émetteur (dans le contexte du marché de taux, les cash flows sont appelés coupons).

condition converge vers l'espérance initiale quand la taille du voisinage s'approche du zéro.

Après un exposé théorique, nous prouvons la vitesse de convergence de nMC et présentons ses avantages sur les simulations Monte Carlo doubles.

6.3.1 Définitions et notations

Les valeurs obtenues sur le i^{ieme} chemin Monte Carlo porteront l'indice ($^{\sharp i}$). Par exemple, $V_n^{\sharp i}$ désignera la valeur espérée en T_n des cash flows entre T_n et T_N sur le i^{ieme} chemin.

- $\mathcal{F}_t$ est la filtration naturelle générée par la source d'incertitude derrière Y_t.
- X_t - l'état du marché en t. Pour simplifier la notation $X_n := X_{T_n}$. Les minuscules représenteront des valeurs réalisées.
- $\phi(x, t, s)$ - la densité conditionnelle de X_t sachant $\mathcal{F}_s$.
- $\Phi([x_a, x_b])$ - la mesure de probabilité associée à $\phi(x, t, s)$.
- $\psi(x, t, s)$ - La fonction de répartition de la variable d'état X_t sachant $\mathcal{F}_s$.
- $\sharp i$ - le i^{ieme} chemin Monte Carlo, i.e., la i^{ieme} réalisation du processus X_t $\sharp i = \{x_0^{\sharp i}, x_1^{\sharp i}, .., x_N^{\sharp i}\}$
- MC - nombre de chemins Monte Carlo
- $\epsilon(x_t)$ rayon du voisinage (contenant les k plus proches voisins) du point x_t.
- $\mathcal{K}(x_t, k)$ - l'ensemble des chemins correspondant aux plus proches voisins de x_t.

6.3.2 Neighbourhood Monte Carlo

Démontrons ici que l'espérance conditionnelle approximative converge vers la valeur exacte, comme annoncé au chapitre 6.3. La quantité $E(V_n \mid X_t \in (x - \epsilon, x + \epsilon))$ peut être estimée de façon efficace en utilisant la formule

$$\widehat{V}_n(x) = \frac{1}{k} \sum_{i \in \mathcal{K}(x)} V_n^{\sharp i} \tag{6.17}$$

où

$$\mathcal{K}(x) \quad = \quad \arg\min_{\epsilon} \{\sharp i \in S, \text{such as } Card\{\sharp i, \left\| x^{\sharp i} - x \right\| < \epsilon\} \geq k\} \tag{6.18}$$

est l'ensemble des k-plus proches voisins de x (i.e. la plus petite boule ouverte contenant x et ayant k éléments). L'ensemble $\mathcal{K}(x)$ est généré par la technique de recherche de k plus proches voisins.

Recherche des k plus proches voisins

Étant donné un ensemble S de MC points et q un point individuel d'un espace d-dimensionnel $\mathbb{R}^d$, on veut trouver les k plus proches voisins de q dans S dans le sens de la métrique Euclidienne. Ce problème est trivial en 1 dimension et relativement facile en 2 dimensions grâce aux diagrammes de Voronoi [85]. Il se complique pour des dimensions plus élevées où la représentation efficace de S est un défi. La recherche des voisins connaît des applications dans la reconnaissances de formes, la compression de données, l'apprentissage automatique et la statistique. La structure de données sous-jacente est une généralisation des arbres k-dimensionnels [43] appelée arbre BBD (balanced box-decomposition tree) [14].

Les arbres BBD réalisent une décomposition hiérarchique de l'espace en plusieurs volumes d-dimensionnels orthogonaux aux axes. Ils ont besoin d'un stockage linéaire dans le nombre des points, et ont la propriété que le nombre des points dans chaque volume décroit de façon exponentielle avec la profondeur de l'arbre. Le volume des partitions décroît exponentiellement aussi. La recherche des k plus proches voisins est possible en temps logarithmique dans cette structure. Le théorème 1 de [14] démontre que cette structure de recherche peut être construite en un temps proportionnel à $d.MC \log MC$. Les détails de l'implémentation sont dans [14]. Une autre option, également intéressante serait de trouver tous les points dans un voisinage de taille ϵ fixe [13].

Convergence de l'estimateur nMC

L'approximation de $E(V_n \mid X_t = x)$ par $E(V_n \mid X_t \in (x-\epsilon, x+\epsilon))$ se fait au moyen d'une intégration Lesbegue sur l'expansion Taylor de la valeur en X_t, et dépend de la distribution de la variable d'état. Pour faciliter la présentation, les résultats suivants supposent que x est unidimensionnel. Ils se généralisent facilement à d-dimensions (la vitesse de convergence aura le même comportement asymptotique en $o(\epsilon)$ que le cas unidimensionnel, mais les coefficients contiendront les dérivés simples, doubles et croisées des composantes de x). V_n est supposé continue, mais ses dérivés peuvent être discontinues en x.

$$V_n(x+dx) = V_n(x) + \frac{\partial V_n}{\partial x}dx + \frac{1}{2}\frac{\partial^2 V_n}{\partial x^2}dx^2 + o(dx^2) \text{ où } dx \in (-\epsilon, \epsilon)$$

L'erreur théorique est estimée à :

$$\zeta(x) = E(V_n \mid X_t \in (x-\epsilon, x+\epsilon) - E(V_n \mid X_t = x) = \int_{-\epsilon}^{\epsilon} \left[z\frac{\partial V_n}{\partial x} + \frac{1}{2}z^2\frac{\partial^2 V_n}{\partial x^2} + o(z^2) \right] \phi(z,t,0)dz \tag{6.19}$$

Pour la distribution uniforme, on obtient :

$$\zeta^{uniform}(x) \quad = \quad cste \times \frac{\partial^2 V_n}{\partial x^2}\epsilon^3 + o(\epsilon^3) \tag{6.20}$$

ce qui est une vitesse de convergence excellente[9]. En augmentant le nombre de chemins Monte Carlo, ϵ peut être rendu arbitrairement petit, et l'erreur de l'estimateur nMC disparait. La taille du voisinage sera considérée constante pour les résultats de cette section, mais nous proposons ensuite une discussion plus générale (ϵ dépendant de la distribution de la variable d'état X_t et du nombre des chemins Monte Carlo). Si la simulation Monte Carlo remplit bien l'espace, la convergence de l'estimateur proposé dépend principalement de la valeur des dérivés du payoff.
L'intérêt de la distribution uniforme est plus qu'académique. Si le voisinage est très petit, toute distribution continue de X_t peut être localement approchée par une distribution uniforme. En conséquence, equation (6.20) donne la vitesse de convergence maximale de l'estimateur nMC. Ceci explique pourquoi il est préférable d'avoir des dérivés petites.

L'erreur de la distribution normale est :

$$\begin{aligned} \zeta^{normal}(x) &= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\epsilon}^{\epsilon} \left[z\frac{\partial V_n}{\partial x} + \frac{1}{2}z^2\frac{\partial^2 V_n}{\partial x^2} \right] \exp\left(-\frac{(z+x)^2}{2\sigma^2} \right) dz \\ &= \epsilon f(x) + o(\epsilon) \end{aligned}$$

[9] Si les dérivés sont discontinues en x, la vitesse de convergence chute à $o(\epsilon^2)$.

La distribution log-normale $\mathcal{LN}(\mu, \sigma)$ montre une erreur proportionnelle à :

$$\begin{aligned}\zeta^{log-normal}(x) &= \int_{-\epsilon}^{+\epsilon} \left[z\frac{\partial V_n}{\partial x} + \frac{1}{2}z^2\frac{\partial^2 V_n}{\partial x^2} \right] \frac{1}{(z+x)\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\ln(z+x)-\mu)^2}{2\sigma^2} \right) dz \\ &= \epsilon g(x) + o(\epsilon)\end{aligned}$$

Les intégrales précédentes contiennent les coefficients $f(x)$ et $g(x)$ dont la forme exacte (assez compliquée) est disponible en utilisant n'importe quel logiciel de calcul symbolique. Les principales distributions rencontrées dans les modèles financiers convergent en ϵ, ce qui est très convenable.

Variance de l'estimateur. L'estimateur dans equation (6.16) est calculé numériquement. Sa variance est la somme de deux facteurs : la variance MC classique et la variance du biais des termes de equation (6.19). Le second dépend de la variance des dérivés premières et secondes et augmente avec la taille du voisinage. Si l'espace est bien rempli (c'est généralement le cas en utilisant quelques dizaines de milliers de chemins), la variance du biais converge à zéro, et disparait. Dans la région où la variance du biais n'est pas infinie, la variance de l'estimateur nMC est environ deux à trois fois supérieure à la variance de la simulation Monte Carlo double, pour le même nombre de chemins. Naturellement, ce ratio se dégrade vite, quand ta taille du voisinage devient trop importante. Les résultats expérimentaux montreront que nMC est plusieurs fois plus rapide que la Monte Carlo double (dMC) pour la même variance.

Les applications cibles, comme la callable Monte Carlo sont relativement tolérantes vis-à-vis de la variance de la valeur de continuation (voir figure 6.5). La figure 6.3 montre que la double Monte Carlo est à préférer pour les applications qui ont besoin d'une grande précision et qui peuvent supporter son coût significatif.

La figure 6.4 montre que la variance du double MC diminue quand le nombre des chemin de la seconde Monte Carlo augmente. La même observation est vraie pour nMC jusqu'à un certain point, après lequel la taille du voisinage devient trop important et l'erreur de l'approximation de l'espérance conditionnelle exacte devient prépondérante.

6.3.3 Convergence globale

Il ne suffit pas de démontrer que l'estimateur nMC converge dans chaque point x. Ces estimateurs ne sont pas indépendants, et il faut assurer que les chemins réutilisés plusieurs fois pour calculer equation (6.17) ne ruineront pas la distribution totale, i.e., la moyenne de estimateurs nMC doit coïncider avec la moyenne des chemins qui sont à la base de leur calcul. Ce n'est pas, a priori, le cas. La mesure de probabilité subit une distortion que nous essayeront de chiffrer en trois étapes.

(a) Premièrement, on calculera la taille (rayon) du voisinage en chaque x, en fonction de la distribution de X_t et du nombre des chemins MC.

$$\Phi(\mathcal{K}(x)) = \int_{x-\epsilon(x)}^{x+\epsilon(x)} \phi(z,t,s)dz = \psi(x+\epsilon(x),t,s) - \psi(x-\epsilon(x),t,s)$$

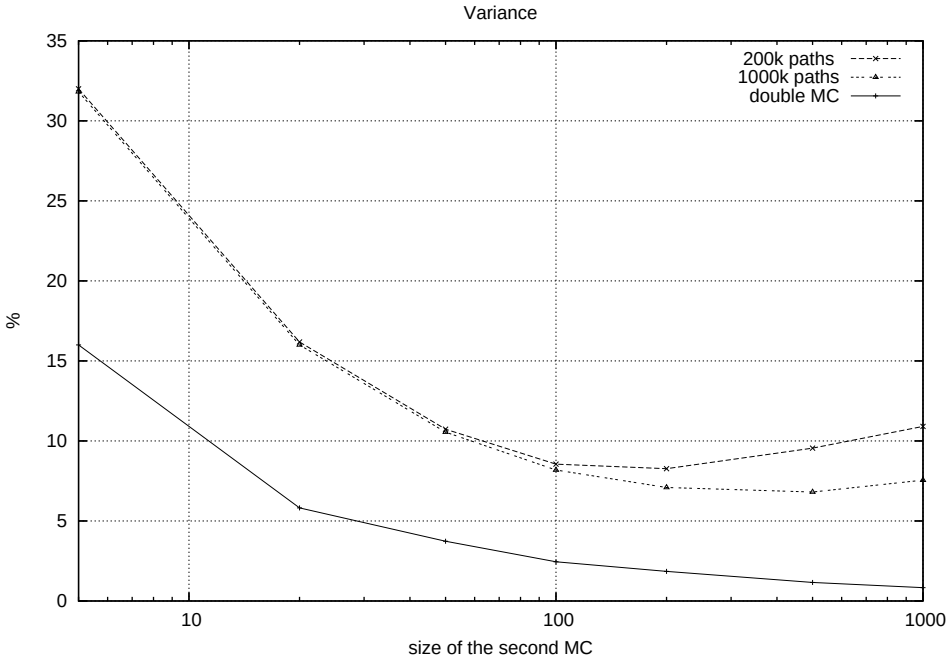


FIG. 6.3 – *Comparaison de l'écart-type de l'erreur commise par nMC et dMC en pourcentage du prix exact. dMC est plus appropriée si besoin d'une grande précision (faible écart-type). L'axe horizontal représente le nombre de chemins du MC interne (pour dMC), ou le nombre de plus proches voisins visités (pour nMC).*

D'autre part, $\mathcal{K}(x)$ contient k chemins sur un total de MC[10]. Les probabilités élémentaires permettent d'écrire :

$$E\left(\Phi(\mathcal{K}(x))\right) = \frac{k}{MC}$$

Or,

$$E\left(\int_{x-\epsilon(x)}^{x+\epsilon(x)} \phi(z,t,s)dz\right) = \frac{k}{MC} \tag{6.21}$$

Dans les cas les plus triviaux, le rayon du voisinage s'obtient analytiquement en inversant equation (6.21). Dans les cas difficiles, il peut être approché comme une fonction linéaire de la densité en x, si $k << MC$. En réarrangeant equation (6.21), on trouve :

$$\frac{k}{MC} = E\left(\int_{x-\epsilon(x)}^{x+\epsilon(x)} \phi(z,t,s)dz\right) \approx 2E(\epsilon(x))\phi(x,t,s) \tag{6.22}$$

et le rayon espéré du k-voisinage de x est :

$$E(\epsilon(x)) \approx \frac{1}{2\phi(x,t,s)} \times \frac{k}{MC} \tag{6.23}$$

[10]Selon equation (6.18), $\mathcal{K}(x)$ peut contenir plus de k chemins, si deux ou plusieurs points se trouvent à distance égale de x.

(b) Ensuite, on exprime la probabilité qu'un chemin appartienne au le voisinage d'un autre chemin. Pour x et y fixés, x est dans le voisinage de y, si et seulement si au moins $MC-k$ points se trouvent à l'exterieur de la plus petite boule ouverte centrée autour de y et contenant x. Par l'indépendance des points, nous avons :

$$\begin{aligned} P\left(x \in \mathcal{K}(y)\right) &= \left\{P\left(\|z-y\| > \|y-x\|\right)\right\}^{MC-k} \\ &= \left\{1 - \int_{\|z-y\|>\|y-x\|} d\phi(z,t,s)\right\}^{MC-k} \end{aligned}$$

Cette expression converge vers zéro quand le nombre de chemins Monte Carlo augmente, puisque de plus en plus de chemins iront s'intercaler entre x et y, diminuant le rayon du voisinage de y, et donc la probabilité que x appartienne à ce voisinage.

(c) Enfin, combien de voisinages contiennent un chemin donné ?

$$\begin{aligned} howmany(x) &= Card\left\{\sharp i, \text{such as } \left\|x - x^{\sharp i}\right\| < \epsilon(x^{\sharp i})\right\} \\ E(howmany(x)) &\approx Card\left\{\sharp i, \text{such as } \left\|x - x^{\sharp i}\right\| < \frac{1}{2\phi(x_t^{\sharp i}, t, s)} \times \frac{k}{MC}\right\} \end{aligned} \quad (6.24)$$

Observons que equation (6.24) peut être évaluée facilement après la simulation, ce qui donne un retour a posteriori sur la qualité de la simulation. Il est cependant possible de donner une expression analytique de l'espérance du nombre des voisinages contenant un chemin donné :

$$\begin{aligned} E(howmany(x)) &= E\left(\sum_{i=1}^{MC} 1_{x\in\mathcal{K}(x^{\sharp i})}\right) \\ &= E\left(\sum_{i=1}^{MC} P\left(x \in \mathcal{K}(x^{\sharp i})\right)\right) \\ &= E\left(\sum_{i=1}^{MC}\left\{1 - \int_{\|z-x^{\sharp i}\|>\|x^{\sharp i}-x\|} d\phi(z,t,s)\right\}^{MC-k}\right) \\ &= \int_{-\infty}^{\infty}\left\{1 - \int_{\|z-y\|>\|y-x\|} d\phi(z,t,s)\right\}^{MC-k} d\phi(y,t,s) \end{aligned} \quad (6.25)$$

Finalement, la distortion de la mesure de probabilité est la différence entre le prix obtenu par la Monte Carlo principale et la moyenne des estimateurs nMC. Le poids d'un chemin dans nMC, est égal au nombre de voisinages dont il fait partie divisé par le paramètre k. Donc,

$$\begin{aligned} distortion &= \frac{1}{MC}\sum_{i=1}^{MC} V_n^{\sharp i} - \frac{1}{MC}\sum_{i=1}^{MC} \frac{howmany(x^{\sharp i})}{k} V_n^{\sharp i} \\ &= \frac{\sum_{i=1}^{MC}\left(\frac{howmany(x^{\sharp i})}{k} - 1\right) V_n^{\sharp i}}{MC} \end{aligned}$$

Afin d'éliminer cette distorsion, on peut appliquer un coefficient correcteur (poids) sur chaque chemin. On peut donc reformuler equation (6.17) :

$$E(V_n \mid X_t \in (x-\epsilon, x+\epsilon)) = \frac{\sum_{i=1}^{k} V^{\sharp i} \times w^{\sharp i}}{\sum_{i=1}^{k} w^{\sharp i}} \tag{6.26}$$

Les poids $w^{\sharp i}$ peuvent être :
- l'inverse du rayon du voisinage
- l'inverse du nombre des voisinages dont le chemin fait partie, comme dans equation (6.25). (Ce choix préserve distribution originale sans biais nMC).

Les tests numériques ont montré que les estimateurs sans corrections sont très proches des estimateurs corrigés. Ceci indiquerait que le coût de calcul de la correction ne serait pas justifié. Intuitivement, on pourrait dire que la plupart des chemins se trouvent dans une région densement peuplée, et assez homogène, avec très peu de distorsion. La distorsion est importante pour les chemins excentrés, mais leur nombre est assez faible.

6.3.4 Calibrage des paramètres de simulation

Si le nombre de chemins Monte Carlo (MC) est fixé, l'erreur de nMC dépend de la taille des voisinages. Celle-ci est une fonction croissante du nombre k de voisins choisis. Si le voisinage est petit, et l'estimateur nMC est sans biais, la variance de l'estimateur de equation (6.17) est proportionnelle à $\frac{1}{\sqrt{k}}$, la vitesse de convergence usuelle de toute méthode Monte Carlo. L'utilisateur est donc confronté à un choix difficile : pour éliminer le biais, il doit choisir un voisinage réduit (donc petite valeur de k), mais, pour réduire l'erreur Monte Carlo, il doit augmenter k. Le compromis idéal dépend des autres données de simulation, i.e., de la fonction payoff et du nombre MC. Il est donc très important de trouver le bon ratio k/MC. La figure 6.4 trace l'erreur de simulation pour plusieurs choix de k et MC.

L'erreur de simulation en t_n est définie comme $\sum_i^{MC} \left|\widehat{V}_n(x^{\sharp i}) - V_n(x^{\sharp i})\right|$, la différence entre l'estimateur nMC et la valeur théorique. Généralement, k doit avoisiner quelques centaines ou milliers, en fonction de la variance de la fonction de payoff. Des valeurs élevées de k sont possibles seulement si le nombre de simulations MC est suffisamment grand.

La figure 6.4 confirme intuitivement les résultats théoriques des sections précédentes : en agissant sur les valeurs de k et MC, l'erreur de simulation peut être rendue arbitrairement petit quand $\epsilon \to 0$, comme l'equation (6.19) le suggère.

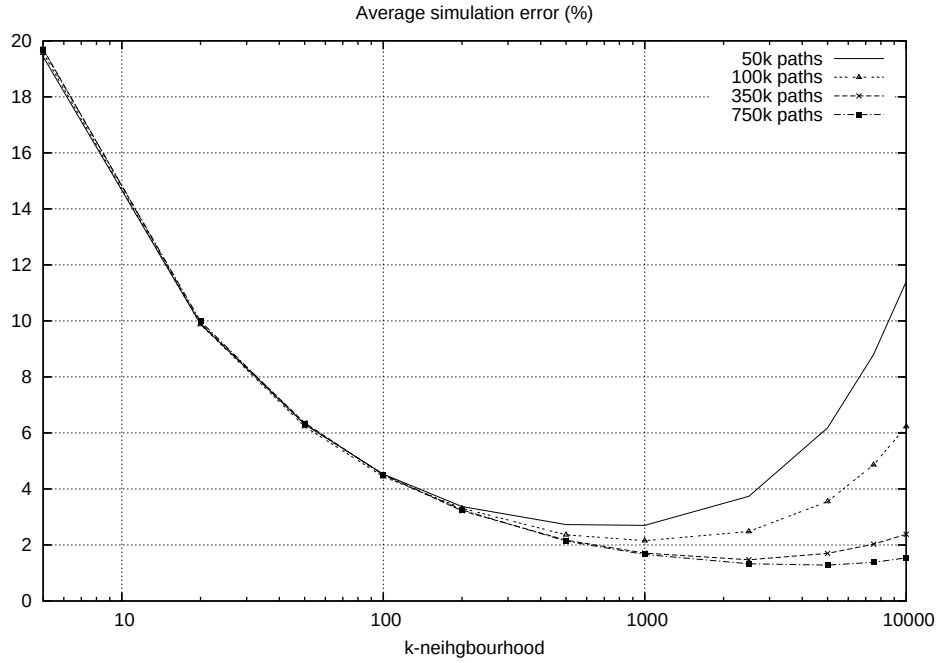


FIG. 6.4 – *Neighbourhood Monte Carlo a calculé la valeur après $t_n = 5Y$. Pas de correction de la densité dans le sens de la section 6.3.3*

6.4 Applications

L'estimateur de valeur résiduelle avec neighbourhood Monte Carlo peut être utilisé en conjonction avec la méthode d'Andersen (voir [12] et section 6.2.3). La méthode LSM a aussi besoin de connaitre la valeur de continuation pour décider d'arrêter ou de continuer un contrat financier. Le calcul de VAR [54], et l'analyse des scénarii de risque peut également bénéficier d'un estimateur conditionnelle pas cher.

La précision de l'estimateur (sa variance et son erreur) a un impact sur la valeur de produit rappelable. La figure 6.5 a été réalisé en multipliant la variable d'état précise par un bruit $(1 + \delta)$. Ici δ est la perturbation (variable gaussienne d'espérance nulle et de variance δ^2).

La figure 6.5 permet de calibrer l'erreur de simulation souhaitée. Après une première chute relativement conséquente, la perte reste très raisonnable pour une incertitude de quelques pourcents dans la variable d'état. Par exemple, réduire l'erreur de simulation de 1% à 0.1% dans nMC aura un impact de moins de 0.01% sur la valeur du produit, mais il multiplierait le temps d'exécution par plus de 100 fois. Ce coût est rarement justifié, et compte tenu de la tolérance des callable Monte Carlo pour une petite erreur dans la valeur de la variable d'état, nMC constitue une alternative viable aux simulations double Monte Carlo.

La table 6.1 compare les temps d'exécution des deux méthodes pour le même niveau de précision.

Les unités de temps ont été normalisées dans la table 6.1 afin de mieux comparer les performances relatives des deux méthodes. nMC est environ 20 fois plus rapide, mais dMC

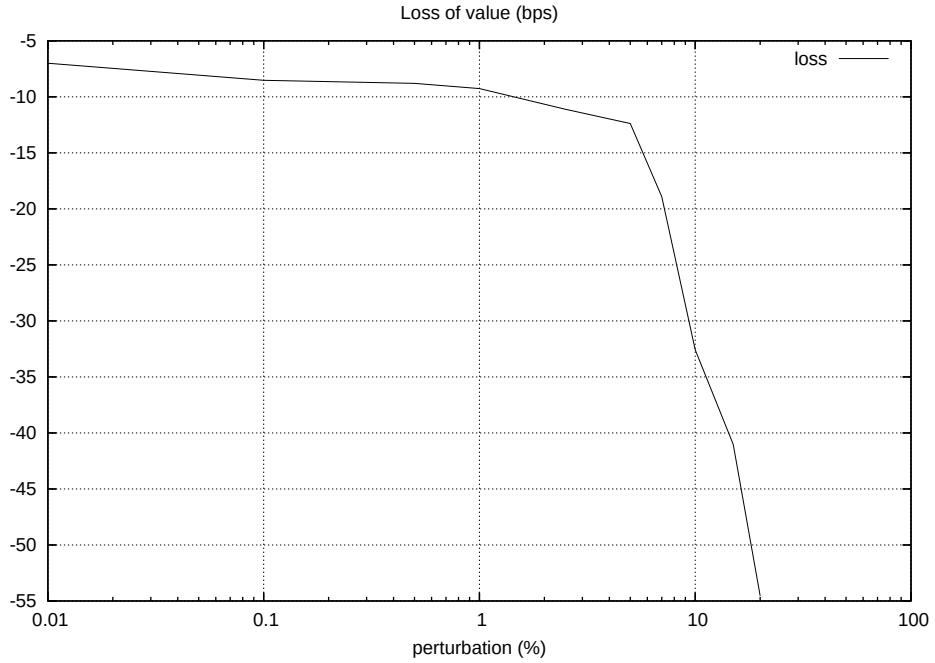


FIG. 6.5 – *La valeur de l'option bermuda décroit quand l'incertitude de la variable d'état augmente. Exemple d'un corridor USD rappelable, maturité 10 ans. Méthode d'Andersen.*

a un meilleur comportement asymptotique, ce qui le rend plus adapté aux applications qui ont besoin d'une grande précision. Le temps d'exécution de nMC a presque triplé, alors que dMC ne prend que deux fois plus de temps pour la même réduction de la variance. Cette observation est consistante avec les autres valeurs expérimentales, et notamment celles de la figure 6.3.

6.5 Conclusion

La méthode neighbourhood Monte Carlo proposée dans ce chapitre semble être une technique très prometteuse pour la valorisation des options bermuda. Son estimateur est une addition très utile à plusieurs algorithmes Monte Carlo américains, comme le très populaire moindres carrés [63] et ses variantes.

Neighbourhood Monte Carlo est destiné aux applications qui ont besoin de connaitre un estimateur conditionnel rapide, mais pas forcément très précis.

Il reste cependant plusieurs pistes de réflexions que nous aimerions regarder dans un futur proche. Le besoin de bien remplir l'espace avec des chemins Monte Carlo est un problème pour les payoffs en dimension élevée. Pire, la recherche des k-plus proches voisins est moins efficace pour des dimensions élevées. En ce moment, la seule solution viable est d'utiliser des techniques de stratification pour regrouper les composantes de X_t ensemble et ainsi diminuer leur nombre. Heureusement, ceci est souvent possible dans la finance.

D'autre part, dans les travaux existants, le nombre de voisins k est le même pour tous

Temps nMC	Temps dMC	stdev nMC	stdev dMC
1	21.8	16.2%	15.8%
1.1	23	10.7%	11.3%
1.3	28.2	8.6%	8.2%
2	35.1	7.6%	7.5%
2.8	39	6.4%	6.8%

TAB. 6.1 – *nMC apporte une performance supérieure aux applications qui n'ont pas besoin d'un estimateur très grande précision, comme c'est le cas de la valeur de continuation dans les callable Monte Carlo. (100k chemins pour nMC, 10Y cap Euribor 6M annulable). La colonne stdev contient l'écart-type de l'erreur de simulation, au sens de la section 6.3.4.*

les estimateurs nMC. Il serait intéressant d'expérimenter un choix de k localement adapté à la variance des payoffs.

Sur le plan algorithmique, la recherche de k-plus proches voisins opère point par point. La construction de la structure de recherche à chaque date de décision, T_i est coûteuse. Comme les structures en T_i sont très similaires à celles en T_{i-1}, on se demande si l'effort dépensé à une date de décision ne pourrait pas être réutilisé plus tard. Travailler sur un algorithme incrémental semble donc être une perspective intéressante. Une fois la recherche des k-plus proches voisins optimisée, nMC bénéficierait alors d'une accélération sous-linéaire.

Enfin, nous avons décrit l'application à la finance qui a motivé notre travail mais le domaine d'application de nMC va bien au delà de ce besoin spécifique. Ainsi, nous trouvons des problèmes du même type s'il s'agit d'estimer la valeur d'une exploitation agricole sachant que les champs peuvent être replantés avant la fin de la saison ou encore l'évaluation des coûts d'usinage d'une pièce qui peut nécessiter un re-réglage de la machine si les premières phases de l'usinage sont insatisfaisantes.

Chapitre 7

Conclusion et perspectives

Les travaux de recherche présentés dans ce mémoire concernent le domaine de la fouille de données (data mining) dans des bases de séquences, et plus particulièrement l'extraction de motifs sous contraintes. Ce type de données séquentielles est présent dans de nombreux domaines d'applications, comme par exemple l'analyse de séquences d'achats dans la grande distribution, les séries financières ou encore l'étude des traces d'usagers sur des sites Web. Le domaine de la finance est un domaine d'application important et il a été au centre de nos travaux. Ainsi, nous avons traité des données financières où l'évolution des cours des actifs est représentée par des séquences d'évènements. Il s'agissait ici de construire des représentations synthétiques de différents scénarii de bourse au moyen de motifs caractéristiques.

L'étude des différentes techniques d'extraction de motifs séquentiels (cf. chapitre 2) a permis d'identifier quatre principales familles d'algorithmes : les approches dites *à la Apriori* dont les principaux représentants sont GSP [3] et PSP [75], celles dites par *Projection* (PrefixSpan [90]), les algorithmes à base de *Listes d'occurrences* comme Spade [107], et enfin une dernière famille exploitant les espaces des versions (e.g., Favst [61]). Chaque famille se compose d'algorithmes adaptés à un contexte particulier, et, clairement, il n'y a pas de méthode universelle (fonctionnant pour des alphabets de taille arbitraire et/ou des dimensions, des densités, ou des contraintes quelconques).

L'intégration des contraintes pour permettre la prise en compte de l'intérêt subjectif et donc les attentes des experts, est à l'origine de nombreux progrès : les motifs extraits doivent satisfaire certaines propriétés spécifiées. Les contraintes peuvent apparaître dans la phase de post-traitement, ou bien être exploitée durant l'extraction. Cette seconde situation a permis de rendre faisable de très nombreuses extractions utiles d'une point de vue pratique (cas typique des motifs fréquents avec intégration de la contrainte de fréquence minimale très tôt dans l'extraction). Ce sont les propriétés de contraintes (anti-monotone, monotone, succincte, préfixe (anti)-monotone) qui déterminent si elle peuvent ou pas être utilisées durant l'extraction (i.e., "être poussées"). Les *expressions régulières* (ER) nous sont apparues comme des contraintes syntaxiques très riches pour de nombreuses applications des motifs séquentiels. De plus, l'étude de l'usage des ER est un cas de conjonction de contraintes anti-monotones et quelconques. Ce problème important et difficile a été placé au cœur de ce mémoire. Après l'analyse des propositions sur la famille Spirit [44], nous avons proposé de nouvelles méthodes pour l'extraction de séquences fréquentes satisfaisant une ER.

Sur le plan théorique, l'étude de l'élagage en la présence d'une ER nous a permis d'identifier la nouvelle famille des *contraintes en cascade*, qui permettent une relaxation localement anti-monotone automatique d'un grand nombre de contraintes courantes. Notre première contribution aux processus de prise de décision est donc de nature algorithmique : nous proposons un nouveau cadre d'extraction sous contraintes de motifs (par exemple des signatures) dans le cas difficile des conjonction d'une contrainte de fréquence minimale avec une contrainte qui n'est a priori ni monotone, ni anti-monotone, ni succincte. Nous avons ainsi proposé la nouvelle famille des *contraintes en cascade*, et l'algorithme générique *Cascaded Constraint Miner (CCM)* pour traiter les extractions qui se laissent décomposer en une série des recherches élémentaires, plus simples et plus faciles à effectuer. L'algorithme Re-Hackle est une implémentation concrète du cadre CCM pour traiter l'extraction des séquences fréquentes satisfaisant une expression régulière. Au cours du développement de Re-Hackle, nous avons proposé un moyen de lier la complexité de la contrainte et les données concrètes pour caractériser l'effort d'extraction. La *cardinalité théorique et expérimentale* introduite au chapitre 4 est un bon moyen pour piloter l'extraction des séquences fréquentes. Les cardinalités des sous expressions d'une contrainte bien partitionnée sont capables de suggérer une stratégie d'extraction locale, adaptée à la décomposition de la contrainte pour plus de performance. L'enjeu est bien de substituer aux techniques d'optimisation ad-hoc des princpes d'adaptativité des stratégies d'élagage. Exploitant le cadre CCM, Re-Hackle est donc l'un des premiers algorithmes adaptatifs, i.e., un algorithme qui arbitre seul entre les différents types d'élagage possibles (élagage sur la fréquence miniamle et élagage par les restrictions syntaxiques spécifiées par les expressions régulières). Dans l'algorithme Re-Hackle, l'espace de recherche est donné par le langage de la contrainte. C'est la contrainte qui génère les candidats et non pas le produit cartésien de sous-séquences fréquentes de la génération antérieure, façon Apriori. Ce travail demande clairement à être poursuivi (études d'autres formes de contraintes, implémentations efficaces, expérimentations multiples sur des cas réels, etc).

Notre seconde contribution aux processus de prise de décision concerne l'amélioration des indicateurs techniques pour l'analyse du marché. Pour celà, nous avons exploité une recherche de signatures à base de motifs, pus précisément le calcul d'ensembles fermés afin de caractériser des scénarii de marché et d'améliorer le rendement des stratégies d'investissement. Il s'agit d'une application aboutie, destinée à motiver le développement d'une nouvelle classe d'outils d'analyse de bourse. Le "back testing" des indicateurs s'est montré capable d'identifier les signatures caractéristiques à chaque période. Elles sont suffisamment stables dans le temps pour pouvoir améliorer les performances de l'indicateur technique durant la période suivante. Les techniques décrites apportent donc un moyen pour améliorer les stratégies quantitatives existantes et ne demandent pas la mise en œuvre de nouvelles stratégies. Dans les exemples traités, la technique des signatures a produit entre 3% à 6% de gains en plus par rapport à l'indicateur seul. Nous avons considéré trois indicateurs : le W%R, le croisement de moyennes mobiles, et un indicateur confidentiel à l'entreprise CIC. L'application à des données haute fréquence apparaît comme une future piste de développement logique. Cette application a également été l'occasion d'une contribution intéressante à la transformation d'une problème d'analyse de données séquentielles dans un problème de calcul d'ensembles fréquents. En effet, les données de départ sont un vecteur à cinq éléments sur quelques jours, donc une série temporelle multidimensionnelle. L'ensemblisation présentée dans ce mémoire permet de transformer ces

données séquentielles et multi-dimensionnelles dans des données transactionnelles simples. Il devient alors possible d'utiliser les très nombreux algorithmes de calculs de motis ensemblistes (e.g., calcul des ensembles fermés et fréquents). Cette utilisation originales des ensembles fermés a permis de faire émerger une caractérisation différente du phénomène sous-jacent aux données du marché.

Notre troisième contribution aux processus de prise de décision concerne l'aide à la valorisation des produits dérivés au moyen d'une optimisation des méthodes de simulation de type double Monte Carlo. Notre méthode baptisée *neighbourhood Monte Carlo (nMC)* semble très prometteuse pour la valorisation des options de rappel. Son estimateur est un ajout très utile à plusieurs algorithmes Monte Carlo américains, comme le très populaire moindres carrés [63] et ses variantes. Cette méthode est destinée aux applications qui ont besoin de connaître un estimateur conditionnel rapide, mais pas forcément très précis. Il constitue encore un projet de recherche en cours. Nous avons prouvé que les estimations qu'il produisait convergeaient vers la valeur exacte. La puissance de calcul nécessaire pour obtenir une estimation peu bruitée (sans toutefois très précise) est plus faible qu'avec les méthodes existantes, i.e., les méthodes double Monte Carlo. Au moment de l'écriture de ce mémoire, nous avons terminé d'analyser les propriétés de l'estimateur nMC, mais nous n'avons pas pu encore l'inclure dans la valorisation des produits exotiques complexes du système de production de l'entreprise où nous travaillons. Il serait intéressant de comparer la variance de l'estimateur nMC à la variance de l'estimateur Longstaff et Schwartz, le standard industriel.

Terminons en dégageant quelques perspectives supplémentaires pour notre travail.

Nous pensons que le cadre Cascaded Constarint Miner a un potentiel intéressant pour étudier son application à des contraintes de similarité (forcer la similarité avec des motifs de référence, travailler à des notions d'occurrences floues et des contraintes de fréquences "molles" [82, 79, 80]). Re-Hackle réalise un partitionnement implicite de l'espace de recherche. Il serait intéressant de combiner l'arbre Hackle avec des bases de données projetées [49, 90]. Nous pensons qu'une telle fusion des techniques pourrait être plus performante que chacune des méthodes d'origine. Il faut continuer de chercher des voies pour réduire le coût des produits cartésiens dans Re-Hackle. Peut être une combinaison des projection avec l'AST serait une solution efficace comme une alternative aux contraintes préfixe-anti-monotones.

Les perspectives dans le domaine de la finance sont toutes aussi intéressantes. Les améliorations des indicateurs techniques sont statistiquement significatives, et notre méthode est prête à rejoindre les stratégies d'investissement des gestionnaires d'actifs et traders au compte propre. Il est cependant possible de regarder une infinité d'indicateurs techniques, et de chercher une caractérisation des indicateurs pour lesquelles la méthode ne fonctionne pas bien (e.g., les indicateurs à mémoire longue). Nos travaux ont visé l'amélioration des indicateurs pour une stratégie d'investissement figée. Une extension utile serait de rendre la stratégie dynamique, et d'utiliser la recherche des motifs séquentiels ou ensemblistes pour identifier des stratégies conjointement à l'optimisation des indicateurs techniques.

Bien que le *neighbourhood Monte Carlo* soit une méthode puissante, ils restent encore plusieurs pistes de réflexions, que nous aimerions poursuivre. Le besoin de bien remplir l'espace avec des chemins Monte Carlo est un problème pour les payoffs en dimension élevé. Pire, la recherche des k-plus proches voisins est moins efficace en dimension élevée.

En ce moment, la seule solution viable est d'utiliser des techniques de stratification pour regrouper les composantes de la variable d'état ou de diminuer leur nombre. Heureusement ceci est souvent possible dans la finance. Sur le plan algorithmique, la recherche de k-plus proches voisins opère point par point. La structure de données construite à une date de décision est très similaire à la structure qui a été construite à l'étape précédente. Cette construction est très couteuse, et on se demande si l'effort dépensé à une date de décision ne serait-il réutilisable plus tard. C'est un défi de recherche algorithmique excitant. Une fois que la recherche des k-plus proches voisins sera capable de cette prouesse, notre neighbourhood Monte Carlo mériterait d'être revisité pour le faire bénéficier d'une accélération sous-linéaire par rapport au Monte Carlo double.

Annexe A

Démonstration des principaux théorèmes

L'algorithme d'extraction à granularité variable a été introduit dans le chapitre 4. Il convient d'étudier ses propriétés théoriques et de discuter ses performances à l'exécution.

Observation. L'évaluation des nœuds de type union ne demande aucun accès à la base des séquences, l'information contenue dans les nœuds fils suffit pour former l'union de séquences fréquentes.

Théorème 4. *Aucune séquence fréquente de longueur n décrite par l'expression régulière E n'est générée plus tard que la n-ième passe de la base des séquences.*

Démonstration. Trivial pour le niveau 1. Pour les étapes supérieures, prenons une séquence S fréquente de longueur $n \geq 2$. Soit E la sous-expression régulière correspondant à la contrainte qui peut être de forme :

- $E = S$. L'expression est une séquence atomique, son évaluation demande une passe.
- $E = E_1 \oplus E_2$. Pour qu'une séquence S représentée par E soit fréquente, il faut que toutes ses sous-séquences S_1 et S_2 (correspondant à E_1 et E_2) le soient également. En conséquence, S_1 et S_2 ne sont pas vides, et la concaténation des candidats S_1 et S_2 augmente la longueur de S. (Même raisonnement pour $E = E_1 \oplus E_2 \dots E_m$) En appliquant ce raisonnement de manière récurrente sur les sous expressions de E_1 et de E_2, on voit que la longueur des candidats augmente d'au moins une unité à chaque passe de la base de données. La séquence finale de longueur n sera générée en au plus n passes.
- $E = E_1*$. Rappelons que par l'equation (2.1) la fermeture d'une expression régulière est une somme de concaténations. Retour au 2e cas.

Commentaire. Ce théorème tient seulement avec le flux ascendant des candidats. Autrement il fautle restraindre aux expressions sans fermeture de Kleene.

Théorème 5. ***Complétude et correction.*** *L'algorithme Re-Hackle trouve toutes les séquences fréquentes satisfaisant la contrainte imposée(complétude). Aucune autre séquence n'est extraite par l'algorithme (correction).*

Démonstration. L'arbre de l'expression codifie de manière exacte l'expression E de départ. Cette équivalence est démontrée par exemple dans [104, 94]. L'arbre limite l'espace de

recherche, et toute séquence assemblée sur la base de l'arbre correspondant à l'expression de départ. On peut également partir d'une expression supposée erronée renvoyée par l'arbre et montrer que l'une de ses sous-séquences ne faisait pas partie des feuilles de l'arbre, ce qui conduit à une contradiction.

Supposons que la séquence candidate S renvoyée par l'arbre T ne soit pas légale par rapport à l'expression régulière de départ. La séquence a été générée soit par :

- une séquence atomique correspondant à une feuille de l'arbre initial ; dans ce cas l'arbre ne correspondait pas à l'expression régulière de départ. Contradiction ;
- un nœud de type union. La séquence non légale S provient du sous-arbres T_k de T. On recommence le raisonnement avec T_k et S.
- une fermeture Kleene. La méthode de fonctionnement des fermetures Kleene est principalement identique à l'algorithme Apriori, une erreur est possible seulement si les séquences atomiques de la première génération des candidats n'ont pas été légales. Le sous-arbre T' de la fermeture a donc envoyé une sous-séquence S' de S illégale. Le raisonnement continue en prenant S' et T'.
- une concaténation. S est composé de $S_1 \oplus S_2 \dots S_m$. Pour que S soit illégale il faut que au moins une des sous-séquences la constituant ne soit pas légale. Soit $k_1 \dots k_p$ les indices des sous expressions S_k illégale par rapport à E. Le raisonnement continue pour les sous-arbres correpondatns à $S_{k_1} \dots S_{k_p}$. Les itérations 2, 3 et 4 finiront par indiquer une feuille incorrecte (cas 1), ce qui contredit l'hypothèse de correspondance exacte de l'arbre d'expression à l'expression de départ.

La complétude exige que l'arbre soit exploré de manière complète. L'exploration par niveaux commençant à partir des feuilles terminales parcourt l'arbre entier. L'élagage des nœuds violés coupe seulement les branches qui ne pourront plus donner naissance à aucun candidat.

Un nœud de l'arbre est enlevé seulement s'il est de forme $N = N_1 \oplus N_2 \dots N_m$ et $\exists k(1 \leq k \leq m)$, telle que N_k soit violé. Comme la concaténation en N génère des candidats seulement si tous ses fils renvoient des séquences fréquentes, l'élimination de N est justifiée, car il ne pourra plus générer aucun candidat. L'exploration des autres fils $N_1 \dots N_m$, est inutile, on élague N et le sous-arbre qui lui est associé.

Remarques sur l'anti-monotonie de la contrainte. L'espace de recherche étant restreint par la contrainte, l'anti-monotonie ne garantit plus que toutes les séquences possibles soient générées, testées et élaguées si infréquentes. Par exemple, une chaîne $(abd)(ce)$ pourrait être candidate, même si dc n'est pas fréquente, car dc ne fait pas partie de l'espace de recherche et il n'est pas sous-séquence granulaire de $(abd)(ce)$. Cette observation nous indique que la notion d'anti-monotonie doit être revue et complétée, car elle ne fonctionne que dans l'espace complet composé des combinaisons des symboles initiaux.

Annexe B

Tests de signifiance

Cette annexe présente les données qui ont servi à la réalisation du test t de la validation expérimentale de la technique d'amélioration des indicateurs techniques, présentée au chapitre 5. Chaque ligne du tableau suivant correspond à une année de trading pour une action donnée. (40 actions x 5 ans). Pour chaque période, on a utilisé une stratégie avec remise de fonds, i.e., la stratégie se voit allouer 100 unités de valeur, et le tableau reproduit le nombre d'unités de valeur restant à la fin de la période. Le nombre de transactions est également communiqué.

W%R signé		W%R seul		W%R signé		W%R seul		W%R signé		W%R seul	
perf	nb	perf	nb	perf	nb	perf	nb	perf	nb	perf	nb
100	0	94.28	100	117.2	24	117	26	118.34	22	180.58	124
75.06	84	60.83	166	100	0	178.4	50	66.85	154	52.32	206
95.09	14	126.3	72	194.55	90	219.55	96	99.29	24	94.49	84
134.11	86	136.97	114	230.75	72	230.75	72	91.36	18	166.99	74
103.39	10	99.94	20	209.85	56	209.85	56	-54.98	76	-80.67	124
100	0	117.19	78	130	24	130	24	133.49	30	169.1	92
53.13	134	34.76	144	-8.08	76	-120.92	148	124.85	32	163.61	68
108.36	14	121.74	110	99.22	38	-12.17	130	99.25	28	106.45	54
132.12	74	135.76	108	104.46	18	120.88	64	103.42	40	99.3	62
110.53	40	110.53	42	100.61	20	115.48	76	97.67	84	97.36	120
-14.87	116	-70.54	166	100.11	14	101.43	26	153.41	40	207.49	88
104.66	22	89.8	118	110.22	44	120.08	78	134.31	36	156.62	70
104.45	6	119.21	72	21.79	102	19.35	116	120.89	58	115.39	66
87.91	76	100.67	136	113.72	18	128.98	44	-38.16	66	-95.55	134
96.59	18	87.52	74	126.48	48	140.68	74	104.54	38	91.64	134
26.3	44	-88.07	144	107.93	26	115.93	40	100.16	2	135.54	58
92.12	80	84.05	186	100	0	61.16	104	118.15	18	152.91	66
100.54	4	130.42	98	87.69	102	80.76	134	101.55	14	121.5	44
111.09	30	124.03	100	111.93	20	137.52	78	-287.98	180	-329.42	226
100.34	18	96.63	50	100	0	119.38	60	84.51	48	6.92	150
-19.62	94	-23.1	104	99.96	4	99.96	4	104.81	6	137.59	98

97.08	58	162.43	122	43.12	114	-3.85	142	107.89	36	96.42	96
100	0	122.18	106	91.81	52	89.92	140	104.15	12	105.96	34
135.34	54	189.56	114	100.01	8	119.87	76	100	0	121.25	78
112.4	8	139.3	38	110.83	42	124.59	80	33.16	104	24.69	110
44.23	74	43.35	122	116.82	28	125.61	46	97.56	48	82.73	80
82.13	102	56.4	160	118.89	40	138.05	64	97.24	28	97.65	72
114.93	20	135.8	96	100	0	74.68	50	108.27	28	119.37	60
104.01	104	99.68	136	141.8	52	171.47	80	100	0	180.03	82
107.06	18	104.94	40	100	0	157.82	68	27.18	110	32.85	114
-18	28	-757.95	170	109.3	20	109.3	22	181.54	48	214.09	64
134.75	32	-120.81	140	16.67	126	13.24	144	122.9	74	123.05	80
149.62	20	157.65	106	89.51	62	41.32	156	-86.28	102	-309.83	206
104.3	34	165.45	122	111.99	18	137.46	62	118.89	14	191.87	74
112.05	12	114.65	40	100	0	119.96	80	95.61	40	31.47	168
130.36	22	137.28	26	99.62	4	102.53	20	100	0	105.81	48
114.34	62	104.61	128	100	0	114.42	66	100	0	121.65	74
145.07	72	165.45	90	88.93	92	92.12	98	92.01	90	100.31	94
157.05	58	158.45	60	109.88	34	121.61	70	117.1	62	115.49	82
183.7	16	183.7	16	116.75	34	118.38	38	120.47	98	123.12	110
100	0	146.72	70	406.05	22	409.55	24	151.75	36	153.7	42
137.83	122	138.08	130	100	0	29.5	136	193.23	60	254.47	84
154.05	62	179.86	88	38.9	58	-1.5	114	170.37	64	184.74	86
163.13	92	167.25	106	135.47	32	197.24	76	134.05	52	134.05	52
104.85	24	105.7	30	100	0	44.22	188	182.9	98	182.33	106
100	0	169.8	86	110.8	14	107.5	80	127.9	18	127.9	18
55.91	94	51.31	96	100	0	123.94	110	100	0	45.14	90
132.12	64	161.54	102	80.7	122	78.8	136	-40.41	108	-23.73	164
53.16	114	51.86	120	103.35	46	108.97	88	119.31	16	168.33	66
105.65	8	116	32	105.14	42	108.05	68	100	0	125.66	74
-61.45	106	-132.96	140	103.75	24	105.54	40	115.4	12	115.4	14
169.29	58	68.72	138	100	0	-219.34	198	106.38	46	112.9	80
99.8	2	145.21	54	83.03	54	53.26	162	100	0	107.34	60
121.74	36	126.17	116	102.52	10	126.88	64	100	0	100	0
100.5	12	109.15	32	101.36	62	104.57	140	102.24	32	101.43	86
85.88	104	48.3	120	99.62	36	100.95	82	99.81	8	99.81	8
104.35	48	144.56	104	100	0	52.28	88	100	0	12.59	150
110.32	66	120.06	86	85.9	32	96.91	146	94.21	42	84.23	126
133.24	76	127.89	84	109	20	131.53	94	102.39	24	112.06	76
99.85	16	93.95	22	101.03	98	96.58	124	113.58	26	117.38	32
102.38	6	123.02	74	109.61	26	108.37	52	100	0	100	0
112.85	44	115.62	72	130.26	42	119.86	90	105.41	4	119.87	66
100	0	100	0	53.84	98	66.22	116	19.24	138	16.42	140

94.28	50	94.58	68	131.34	44	154.78	72	100.85	28	86.35	100
99.96	4	99.96	4	123.05	68	123.05	68	106.11	10	118.03	84
100	0	6.57	146	118.05	28	121.55	30	105.55	12	119.96	40
80.12	66	25.97	188	-566	112	-766.4	172				

Amélioration de l'indicateur W%R signé.

Le tableau suivant reproduit les tests effectués sur l'indicateur Boîte Noire qui gagne de l'argent, ce qui n'était pas le cas du Williams%R. (il s'agit d'une fonction complexe et confidentielle de l'entreprise). Les résultats sont encourageants.

BN signée		BN seule		BN signée		BN seule		BN signée		BN seule	
perf	nb	perf	nb	perf	nb	perf	nb	perf	nb	perf	nb
100	0	26.08	132	91.12	20	116.34	122	198.1	52	224.6	94
50.08	64	33.04	182	99.88	56	104.52	102	83.72	100	0	174
114.22	42	200.42	210	123.7	30	142.6	66	93.59	32	67.98	168
100	0	128.7	138	100	24	100	182	98.08	12	117.1	186
137.65	70	141.3	76	85.27	36	0	182	84.97	44	77.63	128
129.38	30	51.42	170	73.63	34	120.68	216	94.47	26	83.96	68
75.44	14	0	166	77.74	34	48.59	174	93.55	26	8.4	174
119.63	22	119.05	162	96.56	8	108.12	112	78.7	16	0	154
90.64	22	82.51	118	100	0	143.32	170	86.05	12	61.1	162
100	0	125.6	62	92.08	82	21.06	184	90.3	10	70.62	100
105.89	74	76.53	186	122.51	46	140.32	206	102.05	4	131.3	84
81.68	40	40.44	170	105.78	52	111.93	130	100	0	49.98	190
88.18	42	115.1	188	112.05	54	124.75	76	86.37	40	0	180
88.35	44	75.89	124	113.15	26	36.33	172	97.69	30	96.24	216
104.68	12	128.89	90	90.23	10	0	208	101.89	8	102.49	160
96.05	26	111.59	184	101.57	10	122.35	210	101.62	18	97.7	80
102.99	102	102.16	152	107.91	38	104.24	164	122.47	136	134.85	174
100	0	100	0	98	24	101.74	80	100	130	100	164
104.74	38	107.4	146	100	0	120.53	180	91.32	46	135.71	190
99.33	4	98.05	10	80.3	106	105.91	182	121.6	56	99.5	146
100.5	16	72.67	166	77.09	84	54.7	176	102	4	75.45	60
90.26	32	65.74	170	104.57	24	121	144	94.29	76	124.98	174
100.51	20	116.41	200	110.95	34	128.27	78	55.41	120	31.59	144
103.78	54	114.36	146	100.93	38	51.58	210	140.67	84	144.14	210
103.96	18	123.48	92	101.96	16	70.95	172	104.33	108	116.43	170
77.18	124	73.29	196	98.96	10	106.89	198	105.57	68	101.22	92
95.67	28	97.03	194	93.07	14	69.91	114	87.07	30	87.63	176
104.01	22	143.56	216	104.61	16	111.57	110	93.05	22	54.4	194
100	0	125.69	178	96.11	40	69.53	174	96.59	16	202.82	210
97.24	6	97.24	6	86.06	24	61.84	164	100	0	53.83	140
88.05	10	0	126	108.53	24	153.78	208	115.6	28	144.95	92

94.4	24	13.41	146	100	0	88.29	150	100	0	0	128
104.36	10	172.1	150	94.5	14	94.36	76	93.11	70	58.54	156
100	0	46.68	116	77.7	48	0	188	103.93	50	124.38	146
103.25	14	85.1	72	90.1	22	7.93	184	105.4	22	139.3	164
95.32	8	43.66	166	96.46	24	130.46	162	100	0	99.12	6
101.94	6	59.5	186	98.16	32	121.36	132	70.56	38	11.32	162
106.07	22	121.69	226	110.8	24	139.75	82	92.67	22	33.27	180
94.54	68	94.65	144	120.95	64	130.67	184	98.77	20	115.96	192
102.48	20	115.74	108	100	122	100	168	108.52	28	140.6	146
91.19	70	94.88	166	103.58	48	82.89	198	101.82	6	102.34	66
104.14	30	68.37	130	98.5	4	66.33	112	132.5	98	80.18	188
100.89	42	105.71	138	104.75	6	148.05	98	102.4	12	23.37	202
138.81	82	152.52	152	88.18	26	69.58	166	109.57	20	169.97	210
699.55	88	701.75	92	83.37	18	22.02	156	102.16	68	112.26	140
98.62	24	83.65	154	105.33	18	93.43	202	138.8	46	154.4	86
122.72	66	100.13	170	102.87	12	111.49	146	115.21	26	0	174
104.59	60	124.8	178	105.87	54	101.76	84	88.28	46	98.07	200
124.18	112	123.65	150	91.8	20	88.88	186	97.06	8	99.33	212
104.95	40	105.5	48	100.17	32	123.28	186	99.75	2	99.52	142
100	128	100	176	151.01	126	172.31	198	100	0	95.58	80
95.14	44	57.23	194	101.61	62	125.31	152	111.75	40	115.97	168
106.63	28	140.2	238	103.9	52	132.2	98	100	74	100	168
91.35	84	84.26	172	132	18	100	144	115.29	48	129.55	162
97.27	10	92.69	90	73.95	6	0	180	113.92	64	121.57	126
91.52	28	88.01	172	97.43	38	196.22	214	250.4	58	305	96
90.74	30	57.98	208	90	34	78.94	132	108.33	22	38.99	168
101.33	12	115.15	182	97.65	8	130.85	72	77.64	18	54.35	178
101.48	36	108.59	138	108.74	12	78.75	200	107.2	16	134.96	186
100.04	24	97.6	64	102.93	4	59.24	186	128.67	44	129.97	118
99.8	28	126.25	152	101.63	16	155.71	212	98.4	20	89.1	74
66.02	34	0	172	94.85	20	90.69	134	102.07	30	127.23	204
94.16	46	103.15	174	102.38	20	104.16	84	100	0	122.53	188
88.91	32	86.94	138	100	0	84.6	160	100	0	100	0
99.15	6	86.29	68	90	48	71.48	158	100.14	2	119.9	182
100	118	100	164	109.34	42	154.7	152	100	0	98.87	12
103.1	38	81.47	142	94.48	82	121.78	116				

Amélioration de l'indicateur Boite noire signé.

Glossaire

- **actif financier** Un bien (action, obligation, option, future etc.) qui produit une série de payements positifs (aléatoire ou fixe).
- **back testing** Procedure souvent utilisé par le → *chartisme* et les gestionnaires de portefeuille pour regarder comment la stratégies d'investissement qu'il ont conçue aurait pu se comporter dans le passé.
- **behavioural finance** Ou finance comportemental, est une école contestée qui nie la théorie des → *marchés efficients* et affirme que le comportement irrationnel de certains acteurs comme l'excès de confiance, ou la persistance dans les erreurs produit des distorsions dans les valeurs des actifs qui peuvent être exploitées profitablement.
- **chartisme (analyse technique)** Malgré un grand nombre d'articles contestant la viabilité du chartisme, la pratique persiste à la fois chez les investisseurs institutionnels que particuliers.
- **frontière d'exercice** Suite des variables de décision indexée par le temps, qui sépare la région où une option bermuda est exercée de celle ou elle est laissée continuer.
- **marchés efficients** Théorie financière née dans les années 70 qui affirme que toutes les informations historiques sont connues par les acteurs économiques et intégrées dans les prix des actifs. L'implication des marchés efficients est que → l'*analyse technique* ne peut pas fonctionner.
- **option de rappel** Mécanisme très populaire attaché à un nombre de produits financiers qui permet à l'un des parties de terminer le contrat à une séries de dates prédéterminés. Connue également sous le nom d'option bermuda.
- **produit dérivé** un → *actif financier* dont la valeur dépend du niveau d'un autre actif.
- **simulation Monte Carlo** technique de valorisation numérique des → *produits dérivés*, basé sur le théorème centrale limite, consistant à générer un grand nombre de scénarii identiquement distribués afin de calculer une espérance.
- **structuration** Activité des banques visant à créer de nouveaux → *produits dérivés*.
- **trader** L'opérateur de marché est une personne chargée d'interagir avec les bourses pour le compte d'une banque.
- **trading** prise de position (achat ou vente) des → *actifs financiers*.
- **indicateur technique** Fonction des cours et volumes présents et historiques des → *actifs financiers* qui implique qu'une intervention à la bourse à une espérance de gain positif. Les prédictions des indicateurs techniques ne sont pas scientifiquement prouvées.
- **stock picking** - analyse conduisant au choix des → *actifs financiers* dans lesquelles l'investisseur voit un potentiel de plus value.
- **wealth process** - Evolution au cours du temps de la valeur d'un portefeuille d'→ *actifs financiers*.
- **Williams%R** un → *indicateur technique* particulier utilisé par le → *chartisme* pour indiquer si un actif a été sur-acheté ou sur-vendu.

Bibliographie

[1] Achelis, S. B. (2000) *Technical Analysis from A to Z.* 2nd edition, McGraw-Hill, 380p.

[2] Agrawal, R. Srikant, R. (1994) *Fast Algorithm for Mining Association Rule.* Proc. 20th Int. Conf. Very Large Databases (VLDB'94), Chile, pp.487-499.

[3] Agrawal, R. Srikant, R. (1995) *Mining Sequential Patterns.* Proc. 11th IEEE Int. Conf. on Data Engineering (ICDE'95), Taiwan, pp.3-14.

[4] Albert-Lőrincz, H. (2002) *Étude de l'impact des technologies Data Mining sur la diffusion de spots publicitaires interactifs.* Mémoire d'ingénieur, 81p.

[5] Albert-Lőrincz, H. (2002) *Extraction des motifs séquentiels sous contraintes à granularité variable.* Mémoire de DEA, 30p.

[6] Albert-Lőrincz, H. Boulicaut, J.F.(2003) *Mining frequent sequential patterns under regular expressions : a highly adaptive strategy for pushing constraints.* 3rd SIAM Int. Conf. on Data Mining (SIAM DM'03), USA, pp.316-320.

[7] Albert-Lőrincz, H. Boulicaut, J.F.(2003) *A framework for frequent sequence mining under generalized regular expression constraints.* Proc. ACM Knowledge Discovery in Inductive Databases (KDID'03), Croatia, pp.2-16.

[8] Albert-Lőrincz, H. Boulicaut, J.F.(2006) *Amélioration des indicateurs techniques pour l'analyse du marché financier.* Actes des 6ème Journées Francophones Extraction et Gestion de Connaissances (EGC'06), France. Cepadues RNTI-E-6, pp.693-704.

[9] Albert-Lőrincz, H. Wallez, D. (2006) *Tumbling down the double Monte Carlo : Put at work thy neiughbours !* Nomura International plc. Technical report, Ref. albwd20v10, 12p.

[10] Allen, F., Karjalainen, R. (1999) *Using genetic algorithms to find technical trading rules.* Journal of Financial Economics, vol. 51, pp.245-271.

[11] Aluru, S., ed. (2006) *Handbook of Computational Molecular Biology.* Chapman and All/Crc Computer and Information Science Series, 1104p.

[12] Andersen, L. (1995) *A simple approach to the pricing of Bermudan swaptions in the multi-factor Libor Market Model.* Journal of Computational Finance vol. 3, pp.5-32.

[13] Arya, S. Malamatos, T. and Mount, D. M. (2005) *Space-Time Tradeoffs for Approximate Spherical Range Counting.* 16th Ann. ACM-SIAM Symposium on Discrete Algorithms, (SODA'05), pp.535-544.

[14] Arya, S. Mount, D. M. Netanyahu, N. S. Silverman, R. and Wu, A. Y. (1998) *An Optimal Algorithm for Approximate Nearest Neighbor Searching.* Journal of the ACM, vol. 45, pp.891-923.

[15] Ayres, J. Flannick, J. Gehrke. et al. (2002) *Sequential pattern mining using bitmap representations* Proc. 8th ACM Int. Conf. on Knowledge Discovery and Data Mining, (SIGKDD'02), Canada, pp.429-435.

[16] Backer, L. A., Seshadri, M. (2003). *GP-evolved Technical Trading rules Can OutPerform Buy and Hold.* Proceedings of the 6th Int. Conf. on Computational Intelligence and Natural Computing, USA.

[17] Baldi, P. Brunak, S. (2001) *Bioinformatics : The Machine Learning Approach*, 2nd edition. MIT Press, 400p.

[18] Barraquand, J. (1995) *Numerical valuation of high-dimensional European securities.* Management Science vol. 41, pp.1882-1891.

[19] Bayardo, R. J. Jr. (1998) *Efficiently Mining Long Patterns from Databases.* Proc. 1998 ACM Int. Conf. on Management of Data (SIGMOD'98), pp.85-93.

[20] Bayardo, R. J. Jr. Agrawal, R. and Gunopulos, D. (1999) *Constraint-Based Rule Mining in Large Dense Databases.* Proc. 15th Int. Conf. on Data Engineering (ICDE'99), pp.188-197.

[21] Benninga, S., Björk, T., Wiener, Z. (2002) *On the Use of Numeraires in Option pricing.* Working Paper Series in Economics and Finance 484, Stockholm School of Economics.

[22] Black, F., Scholes, M. (1973) *The pricing of options and corporate liabilities.* Journal of Political Economy, vol. 81(3), pp.637-54.

[23] Bonchi, F., Gianotti, F. Mazzanti, A. Predreschi, D.(2003) *ExAMiner : Optimized level-wise frequent pattern mining with monotone constraints.* Proc. 2003 IEEE Int. Conf. on Data Mining (ICDM'03), pp.11-18.

[24] Boulicaut, J-F. Bykowski, A. and Rigotti, C. (2000) *Approximation of frequency queries by means of free-sets.* 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'00), France, Springer-Verlag LNAI 1910, pp.75-85.

[25] Boulicaut, J-F. Jeudy, B. (2000) *Using constraints during set mining : Should we prune or not ?*, Actes des 16e journées Base de Données Avancées (BDA'00), France, pp.221-237.

[26] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984) *Classification and Regression Trees*, Wadsworth, Int. Group, USA.

[27] Broadie, M. Glasserman, P. (1997) *A stochastic mesh method for pricing high-dimensional American options.* Columbia Business School, New York.

[28] Bykowski, A Rigotti, C. (2001) *A Condensed Representation to Find Frequent Patterns.* Proc. 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'01) USA, pp.267-273.

[29] Calders, T., Rigotti, C., and Boulicaut, J-F. (2005) *A survey on condensed representations for frequent sets.* Constraint-based mining and Inductive Databases, Springer-Verlag LNCS volume 3848, pp. 64-80.

[30] Cooley, R. Mobasher, B. Srivastava, J. (1997) *Web Mining : Information and Pattern discovery on the world wide web* Proc. 9th IEEE Int. Conf. on Tools with Artificial Intlligence (ICTAI'97), USA, pp.558-567.

[31] Cooley, R. (2000) *Web Usage Mining : Discovery and Application of Interesting Patterns from Web Data* PhD. thesis, University of Minesota, 180p.

[32] de Raedt, L. Jaeger, M. Lee, S.D. Manilla, H. (2003) *A theory of inductive query answering.* Proc. 2002 IEEE Int. Conf. on Data Mining (ICDM'02), Japan, pp.123-130.

[33] Dunkel, B. Soparkar, N. (1999) *Data Organization and Access for Efficient Data Mining.* Proc. 15th Int. Conf. on Data Engineering (ICDE'99), pp.522-529.

[34] Capelle, M. Boulicaut, J-F. Masson, C. (2002) *Extraction de motifs séquentiels sous contrainte de similarité.* In Extraction et gestion des Connaissances (EGC'02), France, Hermes. pp.65-76.

[35] Chiu, D-Y. Wu, Y-H. Chen, L.P. A. (2004) *An efficient algorithm for mining frequent sequences by a new strategy without support counting.* Proc. 20th IEEE Int. Conf. on Data Engineering (ICDE'04), USA, pp.275-286.

[36] Diop, C. T. Giacometti, A. Laurent, D. Spyratos, N. (2001) *Extraction incrémentale de règles d'association par combinaison de tâches d'extraction.*, Base de données avancées (BDA'01).

[37] Fama, E. (1970) *Efficient Capital Markets : A Review of Theory and Empirical Work.* in Journal of Finance, vol. 25(2), pp.383-417.

[38] Fama, E. (1991) *Efficient Capital Markets II.* in Journal of Finance, vol. 46(5), pp.1575-1617.

[39] Fama, E., (1998) *Market Efficiency, Long-Term Returns, and Behavioral Finance.* Journal of Financial Economics, vol. 49(3), pp.283-306 .

[40] Fayyad, M. U. Piatetsky-Shapiro, G. Smith, P. Uthurusamy, R. (1996) *Advances in Knowledge Discovery and Data Mining* AAAI/MIT Press, 611p.

[41] Fisher, J. Heun, V. Kramer, S. (2006) *Optimal String Mining Under Frequency Constraints.* 10th European Conf. on Principles and Practice of Knowledge Discovery in Databases (ACM PKDD06), Springer Verlag LNCS 4213, pp.139-150

[42] Frawley, W. J. Piatetsky-Shapiro, G., Matheus C. J. (1991) *Knowledge Discovery in Databases : An Overview.* Knowledge Discovery in Databases, AAAI Press, pp.1-27.

[43] Friedman, J. H. Bentley, J.L. and Finkel, R.A. (1977) *An algorithm for finding best matches in logarithmic expected time.* ACM Transactions on Mathematical Software, vol. 3, pp.209-226.

[44] Garofalakis, M. Rastogi, R. Shim. K. (1999) *SPIRIT : Sequential Pattern Mining with Regular Expression Constraints.* Proc. 25th Int. Conf. of Very Large Data Bases (VLDB'99), United Kingdom, pp.223-234.

[45] Geman, H., El Karoui, N., Rochet, J. (1995) *Changes of numeraire, changes of probability measure and option pricing.* Journal of Applied Probabilities, vol. 32(2), pp.443-458.

[46] Giacometti, A. Laurent, D. Diop, C. T. Spyratos, N. (2000) *La découverte de règles d'association entre vues : vers un processus d'extraction incrémental.*, Base de données avancées (BDA'00), France.

[47] Glasserman, P. (2003) *Monte Carlo Methods in Financial Engineering.* Springer-Verlag. New York, 596p.

[48] Han, J. Kamber, M. (2000) *Data Mining : Concepts and techniques* Morgan Kaufmann Publishers, USA, 533p.

[49] Han, J. Pei, J. Mortazavi-Asl, B. Chen, Q. Dayal, U. and Hsu, M.-C. (2000) *FreeSpan : Frequent Pattern-Projected Sequential Pattern Mining.* Proc. 6th ACM Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD'00), USA, pp.355-359.

[50] Han, J. Pei, J. Yin, Y. (2000) *Mining frequent patterns without candidate generation.* Proc. ACM Int. Conf. on Management of Data (SIGMOD'00), USA, pp.1-12.

[51] Han, J. Pei, J. Yin, Y. Mao, R. (2004). *Mining Frequent Patterns without Candidate Generation : A Frequent-Pattern Tree Approach.* Data Mining and Knowledge Discovery vol. 8(1), pp.53-87.

[52] Hand, D. Mannila, H. Smyth, P. (2000) *Principles of Data Mining* MIT Press, England, 545p.

[53] Hanson, F. B. (1996) *Techniques in Computational Stochastic Dynamic Programming*, Digital and Control System Techniques and Applications, edited by C. T. Leondes, Academic Press, pp.103-162.

[54] J.P.Morgan/Reuters (1996) *RiskMetrics Technical Document*, http ://www.riskmetrics.com/rmcovv.html, 296p.

[55] Kasai, T. Lee, G. Arimura, H. Arikawa, S. Park, K. (2001) *Linear-time longuest common prefixe omputations in suffix arrays and its applications* Proc. 12th Annual Symposium on Combinatorial Pattern Matching (CPM'01), pp.181-192.

[56] Ko, P. Aluru, S. (2005) *Space efficient linear-time construction of suffix arrays* J. Discrete Algorithms vol.3(2-4), pp.143-156.

[57] Kosala, R. Blockeel, H. (2000) *Web Mining research : a survey* ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD'00 Explorations), vol 2. pp.1-15.

[58] Lamberton, D. Lapeyre B. (1997) *Introduction au calcul stochastique appliqué à la finance* Ellipses, 175p.

[59] Lent, B. Agrawal, R. Srikant R (1997) *Discovering trends in text databases* Proc. 3rd Int. Conf. of Knowledge Discovery in Databases and Data Mining (KDD'97), AAAI Press, USA, pp.227-230.

[60] Leung, C. K-S. Laksmanan, L. V.S. Ng, R.T (2002) *Exploiting succinct constraints using FP-Trees.* ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD'02 Explorations), pp.40-49.

[61] Lee, S. D. de Raedt, L. (2005) *An efficient Algorithm for Mining String Databases under Constraints.* Proc. ACM Knowledge Discovery in Inductive Databases (KDID'04), Springer Verlag LNCS 3377, pp.108-129.

[62] Klemettinen, M. Mannila, H. Toivonen, H. (1999) *Rule Discovery in Telecommunication Alarm Data.* Journal of Network and Systems Management, vol. 7(4) pp.395–423.

[63] Longstaff, F.A., Schwartz, E.S. (2001) *Valuing American options by simulation : a simple least-squares approach.* Review of Financial Studies vol. 14, pp.113-147.

[64] Kum, H.C.M. Pei, J. Wang, W.(2003) *Approxmap : Approximate mining of consensus sequential patterns.* Proc. 3rd SIAM Int. Conf. on Data Mining (SIAM DM'03), USA.

[65] Leleu, M. (2004) *Extraction de Motifs Séquentiels Sous Contraintes dans des Données Contenant des Répétitions Consécutives.* Thèse de docteur, Insa Lyon, 137p.

[66] Leleu, M. Rigotti C., Boulicaut J.F. et al (2003) *Go-Spade : Mining sequential patterns over dayasets with consecutive repetitions.* Proc. Int. Conf. Machine Learning and Data Mining (MLDM'03), Springer Verlag LNCS 2734, Germany, pp.293-306.

[67] Leleu, M. Rigotti C., Boulicaut J.F. et al (2003) *Constraint-based sequential pattern mining over datasets with consecutive repetitions.* Proc. Principles and Practice of Knowledge Discovery in Databases (PKDD'03), Springer Verlag LNCS 2838, pp. 303-314.

[68] Manber, U. Myers, G. (1993) *Suffix arrays : A new method for online string searches.* SIAM Journal of Computing, vol.22, pp.935-948.

[69] Mannila, H. Toivonen, H. and A. Verkamo, A. I. (1995) *Discovering Frequent Episodes in Sequences.* Proc. 1st ACM Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD'95), Canada, pp.210-215.

[70] Mannila, H. Toivonen, H. (1996) *Discovering generalized episodes using minimal occurrences.* Proc. 2nd ACM Int. Conf on Knowledge Discovery and Data Mining (SIGKDD'96), USA.

[71] Mannila, H. Toivonen, H. and Verkamo A. I. (1997) *Discovery of frequent episodes in event sequences.* Data Mining and Knowledge Discovery, vol. 1(3), pp.259-289.

[72] Mannila, H. Toivonen, H. (1997) *Level wise search and borders of theories in knowledge discovery.* Data Mining and Knowledge Discovery, vol. 1(3), pp.241-258.

[73] Manzini, G. Ferragina, P. (2004) *Engineering a lightweight suffix array construction algorithm.* Algorithmica, vol.40, pp.33-50.

[74] Merton, R. (1973) *Theory of rational option pricing* Bell Journal Economics and Management Science, vol. 4(1), pp.141-183.

[75] Masseglia, F. (1998) *The PSP approach for mining sequential patterns.* Proc. 2nd European Symp. on Principles of Data Mining and Knowledge Discovery (PKDD'98), LNAI, France, vol. 1510, pp.176-184.

[76] Masseglia, F. Teisseire, M. Poncelet, P. (2003) *Incremental mining of sequential patterns in large databases.* Data and Knowledge Discovery Engineereng, vol. 46(1), pp.97-121.

[77] Masseglia, F. Teisseire, M. Poncelet, P. (2004) *Extraction de motifs séquentiels : problèmes et méthodes.* Ingénierie des Systèmes d'Informations, vol. 9(3-4), pp.183-210.

[78] Méger, N. (2004) *Recherche automatique des fenêtres temporelles optimales des motifs séquentiels.* Thèse de docteur, Insa Lyon, 120p.

[79] Mitasiunaite, I., and Boulicaut, J-F. (2006) *Looking for monotonicity properties of a similarity constraint on sequences.* Proc. ACM Symposium of Applied Computing (SAC'06), Dijon (F). pp. 546-552.

[80] Mitasiunaite, I., and Boulicaut, J-F. (2006) *bout softness for inductive querying on sequence databases.* Proc. 7th Int. Baltic Conf. on Databases and Information Systems (DB&IS '06), Lithuania. pp. 77-82. IEEE Computer Press.

[81] Parthasarathy, S. Zaki, M. Ogihara, M. Dwarkadas S. (1999) *Incremental and interactive sequence mining.* Proc. 8th Int. Conf. on Information and Knowledge management (CIKM'99), USA, pp.251-258.

[82] Moen, P. (2000) *Attribute, Event Sequence, and event type Similarity Notions for Data Mining.*, PHD thesis, University of Helsinki, Finland.

[83] Musiela, M., Rutkowski, M. (1997) *Martingale Methods in Financial Modeling.*, Springer-Verlag, 530p.

[84] Ng, R.T., Lakshmanan, L. Han J. and Pang A. (1998) *Exploratory Mining and Pruning Optimizations of Constrained Association Rules.* Proc. ACM 1998 Int. Conf. on Management of Data (SIGMOD'98), pp.13-24.

[85] Okabe, A. Boots, B. Sugihara, K. and Chiu S. N. (2000) *Spatial Tessellations, Concepts and Applications of Voronoi Diagrams.* John Wiley and Sons Ltd., 2nd edition.

[86] Oates, T. Cohen, P. R. (1996) *Searching for Structure in Multiple Streams of Data.* Proc. 13th Int. Conf. on Machine Learninng (ICML'96), pp.346-354.

[87] Oates, T. Jensen, D. Cohen, P. R. (1997) *Automatically Acquiring Rules for Event Correlation from Event Logs.* Computer Science Department Technical Report 97-14, University of Massachusetts.

[88] Pasquier, N. Bastide, Y. Touil, R. Lakhal, L. (1999) *Efficient Mining of association rules using closed itemset lattices.* Pergamon Information Systems, vol. 24(1), pp.25-46.

[89] Pei, J. Han, J. Mao, R. (2000) *CLOSET : An Efficient Algorithm for Mining Frequent Closed Itemsets* ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pp.21-30.

[90] Pei, J. Han, J. Mortazavi-Asl, B. Pinto, H. Chen, Q. Dayal, U. and M.-C. Hsu. (2001) *PrefixSpan : Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth.* Proc. 17th IEEE Int. Conf. on Data Engineering (ICDE'01), Germany, pp.215-224.

[91] Pei, J. Han, J. Wang, W. (2002) *Mining Sequential Pattern with Constraints in Large Databases.* Proc. 2002 Intl. Conf. on Information and Knowledge Management (CIKM'02), USA, pp.18-25.

[92] Pei, J. Han, J. Wang, W. (2002) *Constrained frequent pattern mining : a pattern-growth view.* ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD'02 Explorations), vol.4. pp.31-39.

[93] Pei, J. Han, J. (2001) *Mining Frequent Itemsets with Convertible Constraints*, Proc. 17th IEEE Int. Conf. on Data Engineering (ICDE'01), pp.433-442.

[94] Pittman, T. Peters, J. (1992) *The Art of Compiler Design : Theory and Practice.* Prentice Hall, 368p.

[95] Press, W. H. Flannery, B. P. Teukolsky, S. A. and Vetterling W. T. (1992) *Numerical Recipes in C : The Art of Scientific Computing* Cambridge University Press, 2nd edition.

[96] Roddick, J. F. Spilioupoulou M. (2000) *A bibliography of Spatial, Temporal and Spatio-Temporal Data Mining Research.*, ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD'00 Explorations).

[97] Srikant, R. Agrawal R. (1996) *Mining Sequential Patterns : Generalizations and performance Improvements.* Proceedings of the 5th Int. Conf on Extending Database Technology (EDBT'96), France, pp.3-17.

[98] Tobin, J. (1958) *Liquidity Preference as Behaviour towards Risk.*, Review of Economic Studies, vol. 25(2), pp.65-86.

[99] Ukkonen, E. (1995) *On-line construction of suffix trees.* Algorithmica 14, pp.249-260.

[100] Wang, K. He, Y. Han, J. (2003) *Pushing Support Constraints Into Association Rules Mining.* IEEE Transactions on Knowledge and Data, pp.642-658.

[101] Wang, K. Tan, J. (1996) *Incremental discovery of sequential patterns.* Proc. Workshop Research Issues on Data Mining and Knowledge Discovery (DMKD'96), Canada.

[102] Weiner, P. (1973) *Linear pattern matching algorithm.* Proc. 14th IEEE Symposium on Switching and Automata Theory, pp.1-11.

[103] Wikipedia, the free Encyclopedia on Bioinfomatics (2007), http ://en.wikipedia.org/wiki/Bioinformatics.

[104] Wilhelm, R. Mauer D. (1995) *Compiler Design.* Addison-Wesley, 606p.

[105] Yan, X. Han, J. Afshar, R. (2003) *CloSpan : Mining closed sequential patterns in large databases.* Proc. 3rd SIAM Int. Conf. on Data Mining, USA.

[106] Zaki, M. J. (2000) *Sequence Mining in Categorical Domains : Incorporating Constraints.* Proc. the 7th Int. Conf. on Information and Knowledge management (CIKM'00), USA, pp.422-429.

[107] Zaki, M. J. (2001) *SPADE : An Efficient Algorithm for Mining Frequent Sequences.* Machine Learning Journal. vol. 42(1,2) pp.31-60, Version préliminaire publié dans les actes de CIKM'99.

[108] Zaki, M. J. , Hsiao, C. (1999) *Charm : An Efficient Algorithm for closed association rule mining* In tech. rep. 99-10 Computer Science, Rensellaer Politechnic Institute.

[109] Zheng, Q. Ksu, K. Ma, S. LV, W. (2002) *The Algorithms of updating Sequential Patterns.* Proc. 5th International Workshop on High Performance Data Mining, in conjunction with 2nd SIAM Conf. on Data Mining, USA