Apprentissage et sélection de réseaux bayésiens dynamiques pour les processus online non stationnaires

Matthieu Hourbracq, Pierre-Henri Wuillemin, Christophe Gonzales, Philippe Baumard

▶ **To cite this version:**

Matthieu Hourbracq, Pierre-Henri Wuillemin, Christophe Gonzales, Philippe Baumard. Apprentissage et sélection de réseaux bayésiens dynamiques pour les processus online non stationnaires. Revue des Sciences et Technologies de l'Information - Série RIA : Revue d'Intelligence Artificielle, 2018, Réseaux bayésiens et modèles probabilistes, 32 (1), pp. 75-109. 10.3166/RIA.32.75-109 . hal-03228681

HAL Id: hal-03228681

https://cnam.hal.science/hal-03228681v1

Submitted on 18 May 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Apprentissage et sélection de réseaux bayésiens dynamiques pour les processus online non-stationnaires.

Matthieu Hourbracq [1,2], **Pierre-Henri Wuillemin** [1], **Christophe Gonzales** [1], **Philippe Baumard** [2]

1. *Sorbonne Universités, UPMC Univ Paris 6, CNRS, UMR 7606 LIP6, Paris, France*
   firstName.lastName@lip6.fr

2. *Akheros S.A.S., France*
   firstName.lastName@akheros.com

RÉSUMÉ. *Les réseaux bayésiens dynamiques (DBNs) fournissent un formalisme graphique probabiliste décrivant, à travers des dépendances conditionnelles, des systèmes dynamiques complexes sous incertitude. Dans la plupart des cas, le processus de Markov génératif sous-jacent est supposé homogène, ce qui signifie que ni sa topologie ni ses paramètres n'évoluent au cours du temps. Par conséquent, apprendre un DBN pour modéliser un processus non stationnaire sous cette hypothèse aboutira à de pauvres capacités de prédictions. Dans cet article, nous nous plaçons dans le cas où la non-stationarité du processus s'exprime par la présence de différents modes de comportement. Dans ce contexte, nous proposons un cadre pour l'apprentissage en temps réel des différents modèles probabilistes sous-jacents à chaque mode, sans hypothèses sur leur nombre et leur évolution. Nous montrons la performance de la méthode sur des données simulées. L'application visée est la modélisation et la prédiction d'incongruités pour un Système de Détection d'Intrusion (IDS) en temps réel; c'est pourquoi un grand soin est attaché à la capacité d'identifier précisément les moments de transition.*

ABSTRACT. *Dynamic Bayesian Networks (DBNs) provide a principled scheme for modeling and learning conditional dependencies from complex multivariate time-series data. However, in most cases, the underlying generative Markov model is assumed to be homogeneous, meaning that neither its topology nor its parameters evolve over time. Therefore, learning a DBN to model a non-stationary process under this assumption will amount to poor predictions capabilities. Thus we build a framework to identify, in a streamed manner, transition times between underlying models and a framework to learn them in real time, without assumptions about their evolution. We show the method performances on simulated datasets. The goal of the system is to model and predict incongruities for an Intrusion Dectection System (IDS) in near real-time, so great care is attached to the ability to correctly identify transitions times.*

MOTS-CLÉS : *DBN, ns-DBN, tv-DBN, non-stationnaire, apprentissage, temps réel*

KEYWORDS: *DBN, ns-DBN, tv-DBN, non-stationnary, learning, real time*

Revue d'intelligence artificielle – n° 2-3/2017, 1-30

DOI:10.3166/RIA.28.1-30 © 2017 Lavoisier

1. Introduction

Dans beaucoup de domaines, particulièrement ceux des systèmes d'information ou de la biologie, les processus observés évoluent avec le temps sur plusieurs échelles. Leur état interne change dynamiquement, décrivant des trajectoires complexes. Certains événements ou entités peuvent en influencer d'autres à un moment donné, mais ces corrélations ne tiennent pas nécessairement en toutes circonstances ou pour toujours. Les influences entre entité peuvent ainsi changer ; et tout modèle souhaitant capturer un tel processus, sans pour autant pouvoir observer le mécanisme responsable de tels changements, ne peut être stationnaire : sa structure et/ou ses paramètres doivent évoluer avec le temps aussi. En effet, dans un cadre stationnaire, seul un comportement serait décrit, moyennant toutes les observations.

1.1. Motivation

Dans l'application visée par notre étude, nous souhaitons modéliser le comportement d'un système d'information au sein d'un réseau d'ordinateurs en temps réel afin d'identifier des comportements d'intrusion. Un IDS (*Intrusion Detection Systems*) a pour but de déterminer parmi les traces des différents trafics d'information (logs, appels systèmes, paquets réseaux, etc.) des classes de comportements anormaux qui peuvent donc présenter des risques de sécurité. On distingue deux approches principales pour les IDS : détection d'anomalie et détection d'utilisation incorrecte (Debar *et al.*, 1999).

La détection d'anomalie modélise les observations sur une période de temps sans intrusion afin de pouvoir observer des déviations par rapport à cette modélisation et de les caractériser comme anomalie. Ses principaux défauts sont la nécessité de la connaissance d'une période sans intrusion et l'absence de remise en cause de cette connaissance, d'adaptation (Debar *et al.*, 1999).

La détection d'utilisation incorrecte, quant à elle, s'appuie sur une base de connaissance de comportements d'intrusion (bases de *malware*, de virus, etc.) sous forme de signatures dans les observations. Les nouvelles observations sont alors comparées à ces signatures. Ces principaux défauts sont également le manque d'adaptabilité à de nouvelles formes d'intrusion (attaque polymorphique ou vulnérabilité *zero-days*, etc.).

En général, les deux approches sont considérées comme complémentaires. Avec l'accroissement exponentiel des réseaux d'entreprises, des *data-centers* qui produisent de très importantes quantités d'observations, ces méthodes requièrent trop d'interventions humaines pour pallier à leurs faiblesses. La constante augmentation en nombre et en complexité des attaques suggèrent le besoin de nouvelles méthodes d'investigation.

1.2. Cadre général

Nous nous plaçons dans un cadre d'apprentissage non supervisé pour les IDS, ce qui invalide les 2 approches précédentes. Il semble raisonnable de supposer que les

processus qui peuvent être observés dans un tel système ont un caractère non stationnaire. En effet, n'importe quel programme ou application peut accepter un large éventail d'entrées, communiquer avec d'autres programmes, et les chemins d'exécution choisis (où le processus est localisé et ce qu'il y fait) sont souvent au moins dépendant des entrées.

Nous avons ainsi besoin d'un cadre général qui permette l'apprentissage de tels processus non-stationnaires en temps réel. L'objectif est d'identifier depuis un flux d'observations les différents comportements que peut suivre le processus pour ensuite les caractériser. Il est important de pouvoir déterminer, par exemple, le caractère récurent, exceptionnel ou anormal d'un comportement identifié. Nous appellerons modes les différents comportements que peut adopter le processus. Dans cet article, nous proposons de nous focaliser sur les réseaux bayésiens dynamiques non-stationnaires. Ainsi, nous souhaitons produire une collection *minimale* de modèles probabiliste représentant chacun un mode et expliquant au mieux la dynamique des processus observés.

L'hypothèse de modes de comportements semble raisonnable (tout du moins plus vraisemblable que l'existence d'un mode unique). S'il n'existe pas de modes de comportements, le système change à tout instant de dynamique et il ne serait donc pas possible de le modéliser. Dans la littérature sur les séries temporelles, notamment en économétrie (Perron, 1989 ; Zivot, Andrews, 2012), cela peut être dû à la présence d'une tendance (déterministe ou stochastique) avec des incidents (chocs) dont les effets sur les paramètres sont temporaires ; ou à l'existence d'une racine unitaire (*unit root*) avec des chocs aux effets permanents.

Les réseaux bayésiens dynamiques (Dean, Kanazawa, 1989 ; Murphy, 2002) sont un formalisme graphique probabiliste décrivant, à travers des dépendances conditionnelles, des systèmes dynamiques complexes sous incertitude. Pourtant, le terme *dynamique* dans les DBNs fait référence à l'évolution du système au cours du temps et non à la dynamique de la structure du réseau ou de ses paramètres. Une fois déterminées (par exemple par apprentissage sur un ensemble d'observations), les indépendances conditionnelles et les paramètres sont censés expliquer la dynamique du comportement du système à tout moment. Toutefois, dans beaucoup d'applications (davantage lorsque les données ne sont pas produites de manière contrôlée) supposer l'homogénéité du modèle sous-jacent décrivant l'état du système semble être une hypothèse trop forte. Ce problème a reçu un attention académique particulière ces dernières années, donnant naissance, par exemple, aux réseaux bayésiens dynamiques non-stationnaires (ns-DBN) (Robinson, Hartemink, 2009 ; 2010 ; Grzegorczyk, Husmeier, 2009 ; 2011 ; Gonzales *et al.*, 2015) ou réseaux bayésiens dynamiques à temps variable (TV-DBN) (Song *et al.*, 2009), avec des applications pour les systèmes biologiques (Grzegorczyk *et al.*, 2008). Puisque les processus ont beaucoup de chemins d'exécutions et un espace d'entrée important, il semble malavisé de supposer qu'un modèle homogène puisse capturer avec précision leur évolution. Deux invocations différentes peuvent donner lieu à deux traces d'exécution totalement différentes. Nous construisons par conséquent notre *framework* sur les ns-DBNs.

Dans cet article, nous proposons un nouvel algorithme pour modéliser les processus non-stationnaires en utilisant les réseaux bayésiens dynamiques non-stationnaires. Après une introduction succincte des (D)BNs et ns-DBNs, nous construisons un algorithme d'apprentissage pour ns-DBNs et présentons un cadre complet d'identification des différents modes d'un comportement non-stationnaire, avant d'évaluer ses performances sur un des cas simulés. Enfin, nous concluons et suggérons des travaux futurs.

2. Réseaux bayésiens dynamiques (non-stationnaires)

2.1. Réseaux bayésiens dynamiques

Les DBNs sont des réseaux bayésiens classiques (Pearl, 1988) dans lesquels les nœuds $\{X_i(t), i = 1 \dots n\}$, représentant des variables aléatoires (discrètes), sont indexés par le temps discret t. On note $\mathbf{X}(t)$ l'ensemble des variables aléatoires au pas de temps t (qu'on appelle par convention le *time slice* t). Les DBNs fournissent une représentation factorisée de la loi jointe P sur un intervalle de temps fini $[1, \tau]$, définie comme suit:

$$P(\mathbf{X}(1) \dots, \mathbf{X}(\tau)) = \prod_{t=1}^{\tau} \prod_{i=1}^{n} P\left(X_i\left(t\right) \mid \mathbf{U}_i\left(t\right)\right) \tag{1}$$

où $\mathbf{U}_i(.)$ dénote les parents de $X_i(.)$ et $P\left(X_i\left(t\right) \mid \mathbf{U}_i\left(t\right)\right)$ dénote la fonction de probabilité conditionnelle associée à la variable aléatoire $X_i(t)$ sachant $\mathbf{U}_i(t)$. $\mathbf{X}(t) = \{X_1(t), \dots, X_n(t)\}$, est appelé une "slice" et représente l'ensemble des variables indexées par le même temps t. Cette probabilité jointe $P(\mathbf{X}(1), \dots, \mathbf{X}(\tau))$ représente les croyances sur les trajectoires possibles du processus dynamique $\mathbf{X}(t)$.

Les DBNs suivent la propriété de Markov d'ordre un ; les parents d'une variable à la *time slice* t doivent se trouver soit dans la *time slice* $t-1$ soit t, autrement dit seul le dernier état passé est necéssaire pour prédire le présent:

$$\mathbf{U}_i(t) \subseteq \mathbf{X}(t-1) \cup \mathbf{X}(t) \backslash X_i(t) \tag{2}$$

De plus, les probabilités conditionnelles sont temporellement invariantes (*propriété d'homogénéité temporelle*) ; ces probabilités dépendent du temps passé et non absolu:

$$P\left(X_i\left(t\right) \mid \mathbf{U}_i\left(t\right)\right) = P(X_i(2)|\mathbf{U}_i(2))) \text{ ,} \forall t \in [2, \tau] \tag{3}$$

Ainsi, pour spécifier un DBN, il est seulement nécessaire de définir la topologie intra-slice (au sein d'une *time slice*), la topologie inter-slice (entre deux *time slice*), ainsi que les paramètres (*i.e* les probabilités conditionnelles, voir Équation (3)), pour les deux premières *time slice*. Le résultat est un 2TBN comme montré en Figure 1.

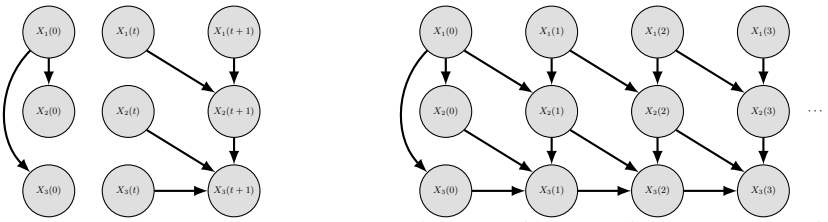

Figure 1. Un 2 Time Slice BN (2TBN) et le réseau bayésien dynamique (déroulé) associé.

Dans cet article, nous considérons que les variables aléatoires $X_i(t)$ sont discrètes et P^t_{ijk} est la probabilité que $X_i(t) = k$ pour l'instanciation j de ses parents, *i.e.*

$$P^t_{ijk} = P\left(X_i(t) = k \mid \mathbf{U}_i(t) = j\right), \quad \begin{array}{l} i = 1, \ldots, n \\ j = 1, \ldots, c_i \\ k = 1, \ldots, r_i \end{array} \tag{4}$$

où r_i est le nombre de valeurs que le nœud $X_i(t)$ peut prendre et c_i le nombre d'instanciations distinctes de $\mathbf{U}_i(t)$.

L'apprentissage d'un DBN $\mathcal{B} : (\mathbf{\Theta}^*, \mathcal{G}^*)$ de paramètres $\mathbf{\Theta}^*$ et structure $\mathcal{G}^*$ depuis une base de donnée $\mathcal{D}$ revient à résoudre $\mathcal{G}^* = \arg\max_{\mathcal{G}} P(\mathcal{G} \mid \mathcal{D})$, avec $P(\mathcal{G} \mid \mathcal{D}) \propto P(\mathcal{D} \mid \mathcal{G})P(\mathcal{G})$. $P(\mathcal{G})$ permet d'imposer des contraintes sur la structure du graphe et est autrement laissé uniforme. $P(\mathcal{D} \mid \mathcal{G})$ est la vraisemblance marginale et est définie par les paramètres $\mathbf{\Theta}_i = P^t_i = P(X_i \mid \mathbf{U}_i, \mathbf{\Theta}_i)$:

$$P(\mathcal{D} \mid \mathcal{G}) = \int P(\mathcal{D} \mid \mathbf{\Theta}, \mathcal{G}) P(\mathbf{\Theta} \mid \mathcal{G}) d\mathbf{\Theta} \tag{5}$$

L'équation (5) peut se résoudre de manière approchée par l'utilisation d'un score tel que BIC (Schwarz, 1978), AIC (Akaike, 1998) ou MDL (Rissanen, 1978) qui cherche à compresser les données $\mathcal{D}$, ou de manière exacte en utilisant des distributions a priori et posteriori conjuguées (les distributions de probabilité a priori et a posteriori sont de la même famille) tel que Dirichlet et le score *Bayesian-Dirichlet* (BDeu).

L'équation (5) devient, avec un a priori de Dirichlet (Heckerman *et al.*, 1995) sur $\mathbf{\Theta}$:

$$P(\mathcal{D} \mid \mathcal{G}) = \prod_i^n \prod_j^{c_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij} + \alpha_{ij})} \prod_k^{r_i} \frac{\Gamma(N_{ijk} + \alpha_{ijk})}{\Gamma(\alpha_{ijk})} \tag{6}$$

avec α_{ijk} l'a priori sur N_{ijk}.

La résolution de cette équation se fait souvent par heuristiques pour diminuer l'espace de recherche, tel que l'algorihme *greedy hill climbing* (Chickering *et al.*, 1995) où un arc est modifié et le score calculé à nouveau jusqu'à ce qu'aucun mouvement n'améliore le score.

Les DBNs ont été appliqués dans des domaines variés tels la reconnaissance vocale (Mitra *et al*., 2011), la détection de pannes (Lerner *et al*., 2000), le diagnostic médical (Charitos *et al*., 2009) ou la biologie (Sicard *et al*., 2011) mais leur application dans le cadre des Systèmes de Détection d'Intrusion est rare (An *et al*., 2006). Cependant, les modèles de Markov (cachés) ont été extensivement proposés pour modéliser des traces d'appels système et de commandes shell (Yeung, Ding, 2003 ; Zanero, Serazzi, 2008), ou du flux réseau (Ourston *et al*., 2003). Dans ce domaine, les réseaux bayésiens sont principalement utilisés de manière statique et souvent à des fins de classification ou de mécanisme décisionnel agrégeant les résultats de modèles plus petits, qui offrent un condensé des données d'entrée (Kruegel *et al*., 2003 ; Mutz *et al*., 2006).

Une variante des DBNs, les *Continuous Time Bayesian Networks* (CTBNs), ont été utilisé pour modéliser du trafic réseau (Xu, Shelton, 2008). Les CTBNs tirent parti du temps continu pour résoudre le problème de granularité temporelle des DBNs qui nécessitent un intervalle de temps constant entre deux *time slice*, ce qui peut les rendre inefficaces en temps de calcul lorsque l'on considère de longues périodes "d'inactivité" ou des observations irrégulièrement espacées temporellement. Le principal désavantage de ce modèle est que deux variables ne peuvent changer d'état simultanément et qu'un paramètre doit être choisi pour définir l'échelle de temps durant laquelle des corrélations peuvent exister.

Le travail dans Xu et Shelton (2008 ; 2010) est proche du nôtre dans leur approche : l'utilisation d'une variable cachée permet de modéliser l'état inconnu de la machine - la structure du réseau est spécifiée manuellement et n'évolue pas. Après apprentissage, une fenêtre glissante est utilisée pour calculer la vraisemblance des observations sous le modèle. Sont annotées anormales toutes observations dont la vraisemblance est sous un certain seuil. Cependant, le nombre d'états de la variable cachée doit être connu à l'avance (deux dans l'article) et de nouveaux modèles ne peuvent être découverts à la volée. En outre, il n'y a aucun mécanisme de raffinement lorsque les fenêtres chevauchent des observations de différents modèles, que le seuil de vraisemblance soit dépassé ou non. Il est fort probable, lorsqu'une fenêtre chevauche une transition, qu'une partie des données à la suite de cette transition ne compense pas suffisament la (haute) vraisemblance à gauche, ce qui donne une vraisemblance globale acceptable et introduit du bruit dans les modèles, qui à long terme rend les détections futures plus difficiles.

2.2. Réseaux bayésiens dynamiques non-stationnaires

Un ns-DBN propose de considérer une homogénéité par période du modèle dynamique. Il est donc représenté par une famille de DBNs $\mathcal{B} : (\mathbf{\Theta}, \mathcal{G})$ de structure $\mathcal{G}$ et de paramètres $\mathbf{\Theta}$ organisés par périodes (ou époques) $\mathcal{T}$:

$$nsDBN = \{(\mathcal{B}_m : (\mathbf{\Theta}_m, \mathcal{G}_m), \mathcal{T}_m)\}_{m \in \mathcal{M}}$$

L'ensemble des modèles $\mathcal{M}$ est fini à tout temps t donné et grandit avec le temps. Pour tout $(m_i, m_j) \in \mathcal{M}^2$ on a $D(\mathcal{B}_{m_i}, \mathcal{B}_{m_j}) > \epsilon$, avec D une fonction de la famille

des f-divergences (comme la distance de Hellinger (Beran, 1977) ou la divergence de Kullback-Leibler (Kullback, Leibler, 1951)). En effet il n'existe pas deux modèles dans $\mathcal{M}$ ayant la même dynamique. La complexité des algorithmes d'apprentissage des ns-DBNs dépend de la cardinalité de $\mathcal{M}$ (d'autant plus lorsque le nombre de transitions et leur temps d'arrivé ne sont pas connus).

Il n'existe pas encore de cadre pour modéliser la dynamique des transitions pour les ns-DBNs. En conséquence, les ns-DBNs représentent des processus non-stationnaires en supposant une stationnarité par morceau (par époque), ce qui semble être une hypothèse raisonnable. On suppose aussi que la fréquence d'observations du processus est nettement supérieure à sa fréquence de changement de dynamique. Ils héritent en outre des avantages et inconvénients des DBNs.

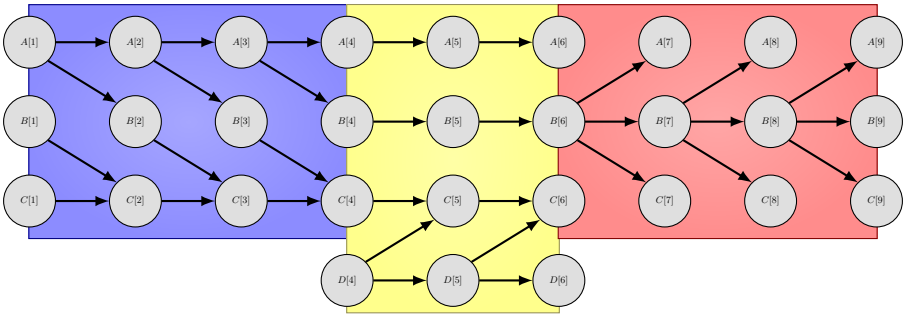

Figure 2. Réseau bayésien dynamique non-stationnaire (ns-DBN) avec 3 différentes époques. Des DBNs d'époques différentes peuvent avoir des paramètres, des structures et même des variables différentes. Pour des raisons de simplicité, les BNs a priori comme en Figure 1 ne sont pas représentés.

Bien que, dans un ns-DBN, différentes époques peuvent avoir des jeux de paramètres, des structures et même des variables différentes (voir Figure 2), Grzegorczyk et Husmeier (2009) se focalise sur l'évolution des paramètres sous une structure fixe. Dans Robinson et Hartemink (2010), l'accent est mis sur l'évolution structurelle (douce) du réseau. Sous ces approches, le nombre de variables et leurs domaines restent constants avec le temps (même si certaines ne sont pas observées au cours d'époques entières). Gonzales *et al*. (2015) ont une approche similaire à la nôtre, autorisant la structure, les paramètres ainsi que les variables et leurs domaines à varier au cours du temps. Cependant, nous utilisons des critères différents ainsi qu'un mécanisme pour les fenêtres chevauchant des événements de différents modèles afin de raffiner les moments de transition détectés. En outre, nous souhaitons pouvoir identifier si un modèle est récurrent ou exceptionnel au lieu d'en créer un nouveau après chaque transition détectée.

Les algorithmes d'apprentissage pour les ns-DBNs consistent à identifier les différentes époques et les DBNs associés à chacune. Les algorithmes actuels se concentrent soit sur l'évolution de la structure soit sur celle des paramètres (principalement pour réduire l'espace de recherche du problème). Malheureusement ces algorithmes nécessitent, pour ainsi dire tous, la disponibilité de toute la base de données et ne peuvent

être utilisés dans le cadre de flux de données (apprentissage *online*), en temps réel, sauf peut-être pour un premier apprentissage afin de s'assurer que la première fenêtre de données ne provient pas déjà de plusieurs modèles distincts. Ces méthodes n'offrent pas non plus la possibilité de réutiliser un DBN appris précédemment. L'apprentissage *offline* des ns-DBNs est habituellement réalisé en utilisant un score traditionnel (*i.e.* BDeu) modifié pour prendre en compte les statistiques suffisantes qui ont besoin d'être spécifiées par époque pour trouver les meilleures transitions ainsi qu'un ensemble de mouvement étendu pour l'apprentissage de la structure, qui doit elle aussi être spécifiée par époque.

Ainsi, Robinson et Hartemink (2010) modifient l'équation (6) pour tenir compte des différentes structures $\mathcal{G}_1, ..., \mathcal{G}_m$ séparées par les transitions $\mathcal{T}$, en supposant l'indépendance mutuelle des paramètres d'une époque à une autre et des variables de domaine stable :

$$P(\mathcal{D} \mid \mathcal{G}_1, ..., \mathcal{G}_m, \mathcal{T}) \propto \prod_h^m \prod_i^n \prod_j^{c_i} \frac{\Gamma(\alpha_{ij}(I_h))}{\Gamma(N_{ij}(I_h) + \alpha_{ij}(I_h))} \prod_k^{r_i} \frac{\Gamma(N_{ijk}(I_h) + \alpha_{ijk}(I_h))}{\Gamma(\alpha_{ijk}(I_h))} \tag{7}$$

avec I_h l'intervalle sur $\mathcal{D}$.

L'apprentissage est réalisé par échantillonnage (Grzegorczyk, Husmeier, 2009 ; Robinson, Hartemink, 2010). En effet, lorsque les temps de transitions ne sont pas connus à priori, la distribution a posteriori sur les structures est trop complexe pour un calcul exact (Robinson, Hartemink, 2010). La stratégie d'échantillonnage est différente selon que le nombre de transitions et/ou leur temps d'arrivées sont connus ou non (plus il y a d'incertitude et plus la stratégie est complexe).

Cependant, nous avons besoin de règles plus simples pour atteindre des performances temps réel avec un flux d'observations continu. Gonzales *et al*. (2015) propose une approche intéressante : prendre en compte la force des arcs en utilisant l'information mutuelle d'un nœud et de ses parents et utiliser les paramètres précédents comme a priori pour les nœuds qui ne changent pas d'un modèle à l'autre. Davantage d'examens sont nécessaires puisque l'information peut se propager différemment dans un réseau selon les nœuds observés.

2.3. Granularité temporelle et problèmes de modélisation

Nous avons vu qu'un ns-DBN peut avoir des jeux de structure et paramètres différents d'une époque à l'autre. Cependant, il est aussi possible que les processus sous-jacents n'aient pas la même fréquence d'échantillonnage inter-époques et même intra-époque. Dans certaines applications, telles que la météorologie, la bio-informatique ou la physique, le temps est vraisemblablement une donnée importante et la granularité temporelle est adaptée. Dans beaucoup d'autres contextes, notamment lorsque l'échantillonnage n'est pas sous contrôle (systèmes informatiques, relevés de capteurs) et ne peut pas être synchronisé entre divers équipements, seule la séquence des évènements est importante. Le temps qui sépare une *time-slice* d'une autre n'est pas

constant et n'a que peu de sens ; comment prendre en compte le temps qui sépare deux instructions sur un ordinateur sachant qu'on peut parler de temps CPU (qui ne tient pas compte des opérations d'entrée-sortie ou du *multi-threading*) ou de temps humain (qui fait fi de l'ordonnanceur du système, qui change de processus à la volée). Ceci n'est pas un problème dans notre cas puisque nous sommes davantage intéressés par le déroulement du processus (inter-mode) que par sa temporalité (intra-mode). Comme mentionné dans (Xu, Shelton, 2008), il n'existe pas toujours de procédé justifiable pour intégrer le temps entre évènements dans un DBN selon le domaine d'application.

3. Apprentissage de processus non-stationnaires

Nous présentons dans cette section un nouvel algorithme pour l'apprentissage en ligne de réseaux bayésiens dynamiques non-stationnaires. Dans la littérature (Grzegorczyk, Husmeier, 2009 ; Robinson, Hartemink, 2010 ; Grzegorczyk, Husmeier, 2011), l'hypothèse que deux modèles temporellement adjacents sont gouvernés par des distributions et/ou une structure similaires est souvent faite. Cependant, nous ne ferons pas ici cette hypothèse et ne nous restreindrons donc pas à des évolutions douces d'un modèle à l'autre.

Les données sont reçues de manière continue et remplissent une fenêtre $\mathbf{w}$ de taille r avant d'être traitées. Pour chaque nouvelle fenêtre d'observations, notre algorithme doit choisir entre utiliser un modèle déjà connu ou en créer un nouveau. Ce choix est basé sur la vraisemblance des données fenêtrées. En effet, nous nous attendons à ce que la vraisemblance d'un modèle décroît si le comportement sous-jacent change (voir Figure 3). L'algorithme commence par une phase d'initialisation qui fournit un premier réseau servant de point de départ.

Plus formellement, à tout instant τ, l'algorithme confronte une collection de M modèles (DBNs) $\{\mathcal{B}_m : (\mathbf{\Theta}, \mathcal{G})_m\}_M$ à la fenêtre de données $\mathbf{w}[\tau, \tau+r]$. Nous sommes intéressés par le caractère récurrent ou exceptionnel des modèles et ne pouvons simplement en créer de nouveaux après chaque déviation rencontrée, c'est pourquoi nous avons besoin de savoir s'il existe déjà un modèle m acceptable pour $\mathbf{w}$, potentiellement différent du modèle courant. Par ailleurs, bien évidemment, notre algorithme doit être capable de créer de nouveaux modèles lorsque ceux existants ne décrivent pas correctement les trajectoires observées dans $\mathbf{w}$.

Ainsi, à partir d'un flux d'observation, nous attendons de notre algorithme de fournir : i) une ensemble de dBNs, ii) un ensemble de fenêtres temporelles découpant le flux et iii) pour chaque fenêtre temporelle, une sélection du dBN qui décrit au mieux le comportement du signal dans la fenêtre. On remarque qu'il est alors facile de discriminer entre des dBNs récurrents (caractérisant beaucoup de fenêtres) et des dBNs exceptionnels (caractérisant peu de fenêtres, voire une seule fenêtre).

3.1. Apprentissage avec variables fixes et fenêtre de taille fixe

Soit un ensemble M de dBNs, il s'agit, pour chaque fenêtre, soit de trouver $m^* \in M$ le modèle décrivant au mieux les observations de la fenêtre, soit d'apprendre un nouveau modèle m' qu'on ajoutera alors à la liste des modèles possibles. Dans un premier temps, nous supposons que les variables suivis dans le flux ne changent pas au cours du temps. Nous supposons également que la taille de la fenêtre temporelle est fixe.

Pour évaluer à quel point un modèle m est capable d'expliquer les données $\mathbf{w}[\tau, \tau + r]$, nous utilisons un critère simple basé sur la vraisemblance des données $\mathbf{w}$. Dans un DBN stationnaire, la log-vraisemblance des données vis-à-vis d'un réseau de structure $\mathcal{G}$ et de paramètres $\mathbf{\Theta}$ est :

$$LL(\mathbf{w} : \mathbf{\Theta}, \mathcal{G}) \propto \sum_{t=\tau}^{\tau+r} \sum_{i,j,k} N_{ijk} \log(\theta_{ijk}) \tag{8}$$

où $\theta_{ijk} = P\left(X_i(t) = k \mid \mathbf{U}_i(t) = j\right)$ et N_{ijk} est le nombre de cas où $X_i(t) = k$ et $\mathbf{U}_i(t) = j$ dans $\mathbf{w}$.

Pour chaque DBN $\mathcal{B}_m : (\mathbf{\Theta}_m, \mathcal{G}_m)$, nous calculons donc $LL(\mathbf{w} : \mathbf{\Theta}_m, \mathcal{G}_m)$. Le meilleur modèle $\mathcal{B}_{m^*}$ pour w ne peut cependant pas encore être sélectionné en maximisant LL dans l'Équation (8) puisque l'algorithme peut/doit aussi créer des modèles à la volée.

À cet effet, on peut remarquer que la distribution de la log-vraisemblance d'une fenêtre $\mathbf{w}$ est approximativement normalement distribuée (comme la somme de $r + 1$ variables aléatoires indépendamment et identiquement distribuées, en utilisant le théorème central limite). Nous proposons donc un test d'hypothèses statistique dans le but de trouver la log-vraisemblance p-value LL_{tr} tel que 99% des appariements se font avec une log-vraisemblance égale ou supérieure (voir Figure 3). Pour calculer une première estimation de LL_{tr} après avoir découvert un nouveau modèle, nous utilisons l'échantillonnage de Gibbs (Casella, George, 1992) : des trajectoires sont échantillonnées depuis le DBN, autant que nécessaire pour remplir plusieurs fenêtres, avant de calculer leurs vraisemblances en faisant glisser la fenêtre. Pour chaque modèle $\mathcal{B}_m$, nous calculons donc $\frac{LL_{tr}(\mathbf{\Theta}_m, \mathcal{G}_m)}{LL(\mathbf{w}:\mathbf{\Theta}, \mathcal{G})}$. Notre règle de sélection devient :

$$m^* = \arg\max_{m \in \mathcal{M}_{0.97}} LL(\mathbf{w} : \mathbf{\Theta}_m, \mathcal{G}_m)$$

$$\text{où } \mathcal{M}_{0.97} = \left\{ m \text{ t.q. } \frac{LL_{tr}(\mathbf{\Theta}_m, \mathcal{G}_m)}{LL(\mathbf{w} : \mathbf{\Theta}_m, \mathcal{G}_m)} \geq 0.97 \right\} \tag{9}$$

Le seuil est de 0.97 plutôt que 1 sur le ratio de vraisemblance puisque le premier apprentissage est, selon la taille initiale de la fenêtre, imprécis (peu d'événements observés pour potentiellement beaucoup de paramètres à apprendre) et nous autorisons une certaine divergence de cet ordre de grandeur. Ce seuil pourrait être dynamique et

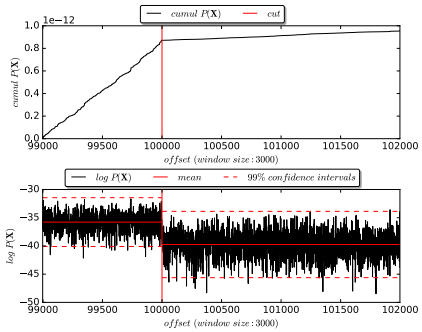

Figure 3. Cumulé de $P(\mathbf{X})$ et $log\ P(\mathbf{X})$ pour une fenêtre chevauchant deux comportements distincts. La ligne rouge verticale représente une transition entre deux modèles et les lignes horizontales les gaussiennes avec des intervalles de confiance à 99%.

autoriser de moins en moins de divergence entre vraisemblance observée et vraisemblance attendue.

Si l'ensemble des modèles $\mathcal{M}_{0.97}$ dans l'Équation (9) est vide, aucun modèle actuellement appris ne représente correctement les observations de la fenêtre courante. L'algorithme apprend alors un nouveau DBN et le sélectionnera comme modèle courant. Par contre, si un modèle $\mathcal{B}_m$ déjà présent est sélectionné, il y a quand même une phase d'apprentissage afin de mettre à jour les paramètres (et éventuellement la structure) au regard des nouvelles données. Nous n'utilisons pas $\mathbf{\Theta}_{m^*_{t-1}}$ ni $\mathcal{G}_{m^*_{t-1}}$ comme a priori sur $P(\mathcal{G}_{m_t})$ laissant la structure, le domain des variables et les paramètres évoluer sans contraintes. En effet, au fur et à mesure que les observations augmentent en quantité pour les modèles, leur structure doit être ré-évaluée : nous choisissons de ré-estimer la structure (et les paramètres) dès que la quantité d'observations a doublé depuis le dernier apprentissage de structure (voir ligne 8 Algorithm 1).

Durant les expériences, la Figure 6 montre comment une fenêtre de mauvaise taille peut perturber l'apprentissage. En effet, si quelques points en fin de fenêtre appartiennent à un autre modèle, leur nombre et/ou effet peut ne pas être suffisamment important par rapport à la taille totale de la fenêtre pour rejeter l'hypothèse nulle. Nous proposons donc, dans un second temps, d'ajouter un mécanisme d'adaptation de la taille de la fenêtre au cours de l'apprentissage.

3.2. Fenêtre dynamique adaptative

Lorsque l'algorithme prédit, pour la fenêtre courante, un modèle $\mathcal{B}_{m^*_t}$ différent du modèle $\mathcal{B}_{m^*_{t-1}}$ de la fenêtre précédente (*i.e.* $\mathcal{B}_{m^*_t}$ est soit un DBN nouvellement créé ou un DBN déjà connu), la prédiction porte non seulement sur le changement de dynamique du processus mais aussi sur le temps τ auquel ce changement survient (le

point de changement). Dans cette section, nous proposons d'investiguer davantage la valeur de ce point en regardant la distribution de la vraisemblance dans $\mathbf{w}[\tau-r, \tau+r]$.

Un point de changement représente une transition d'un état à un autre du processus qui génère ces données. La Figure 3 montre la valeur cumulée de $P(\mathbf{X})$ et $log\ P(\mathbf{X})$ pour une telle fenêtre avec un point de changement ($c = 100\,000$). En supposant les données i.i.d. la série temporelle $log\ P(\mathbf{X}(t))$ est stationnaire et les paramètres du modèles sont donc constants. Si la série n'est pas stationnaire alors elle provient de deux modèles distincts. Pour trouver une valeur correcte de c, on pourrait caractériser le changement de pente dans le cumul de $P(\mathbf{X})$. Cependant, pour être plus précis, nous maximisons sur c la vraisemblance d'un modèle où le point de changement c sépare deux processus stationnaires distincts (ici gaussiens) avec μ_t constant qui ne dépend pas de t, et la fonction de covariance qui ne dépend que de la différence de temps entre deux mesures.

L'algorithme est itératif et garde en mémoire les k dernières fenêtres passées de la distribution de la log-vraisemblance pour un modèle donné (l'effet et le choix de k sont discutés en section 4, Expériences et Résultats). Pour trouver un premier point de changement dans $\mathbf{w}[\tau-r, \tau+r]$, l'algorithme construit une fenêtre D de taille au plus $(k+2)r$ (certaines fenêtres passées peuvent être de taille inférieure à r si un point de changement avait été trouvé). L'algorithme cherche un premier point en maximisant sur $c \in [\![\tau - r, \tau + r]\!]$ la vraisemblance de deux modèles gaussiens, l'un à gauche de c sur $[\![0, ..., kr, ...,\text{c}[\![$ et l'autre à droite sur $[\![c, (k+2)r]\!]$.

La log-vraisemblance des données D sous une loi normale est, lorsque μ et σ sont estimés depuis D :

$$LL(D, \mu, \sigma) \propto -\frac{n}{2} \cdot \big(log(\sigma^2) + log(2\pi) + 1\big)$$

c sépare donc deux processus gaussiens tel que la log-vraisemblance sous ces deux modèles est meilleure que la log-vraisemblane sous un seul modèle:

$$LL(D[0, c[, \mu_1, \sigma_1) + LL(D[c, (k+2)r], \mu_2, \sigma_2) \geq \alpha \cdot LL(D, \mu, \sigma)$$

avec α un paramètre de coût / pénalité pour la création d'un point de changement (l'effet et le choix de ce paramètre sont discutés en section 4, Expériences et Résultats). Un point n'est valide que si la log-vraisemblance moyenne à gauche n'est pas plus basse que la log-vraisemblance moyenne à droite. En effet, nous ne souhaitons pas avoir de point de changement lorsque les données observées sont plus vraisemblables que par le passé.

Soit $f(c) = LL(D[0, c[, \mu_1, \sigma_1) + LL(D[c, (k+2)r], \mu_2, \sigma_2) - \alpha \cdot LL(D, \mu, \sigma)$. À chaque itération, l'algorithme cherche donc c^* tel que :

$$c^* = \arg\max_{c \in \mathcal{C}} f(c)$$

$$\text{avec } \mathcal{C} = \left\{ c \text{ t.q. } \begin{array}{c} f(c) \geq 0 \\ \mu_1 \geq \mu_2 \end{array} \right\} \quad (10)$$

Après avoir trouvé un point de changement, l'algorithme recommence à gauche de ce point (nous avons toujours les fenêtres passées). Nous ne cherchons qu'à gauche puisque nous sommes intéressés par le premier point à partir duquel le signal de log-vraisemblance ne correspond plus au modèle courant. Lorsqu'aucun point n'est trouvé, l'algorithme se termine.

Nous mettons ensuite à jour $\mathcal{B}_{m^*_{t-1}}$ sur $\mathbf{w}[\tau - r, \tau - r + c^*]$ avec le point de changement optimisé c^* le plus à gauche. Si $\mathcal{B}_{m^*_t}$ est un nouveau modèle. Nous utilisons un a priori de Dirichlet non informatif et nous supposons que les paramètres et la structure évoluent sans corrélations d'un modèle à un autre.

Dans la section 4, Expériences et Résultats, nous montrons qu'il est possible de se passer de l'équation (9) et d'utiliser directement l'équation (10). La règle de sélection se base maintenant sur la log-vraisemblance moyenne avant c^* et devient alors :

$$m^* = \arg\max_{m \in M_c} \frac{LL(\mathbf{w}[0, c^*_m[: \mathbf{\Theta}_m, \mathcal{G}_m)}{c^*_m}$$

$$\text{avec } M_c = \left\{ m \text{ t.q. } \begin{array}{lcl} \mathcal{B}_{m_t} = \mathcal{B}_{m^*_{t-1}} & \Rightarrow & c^*_m \geq 0 \\ \mathcal{B}_{m_t} \neq \mathcal{B}_{m^*_{t-1}} & \Rightarrow & c^*_m \geq c_{min} \end{array} \right\} \quad (11)$$

Si $\mathcal{B}_{m_t} = \mathcal{B}_{m^*_{t-1}}$ il nous faut alors $c^*_m \geq 0$ ce qui signifie qu'on peut avoir ou ne pas avoir un point de changement pour le modèle courant. Si $\mathcal{B}_{m_t} \neq \mathcal{B}_{m^*_{t-1}}$ alors $c^*_m \geq c_{min}$; il ne peut y avoir de point de changement pour un autre modèle $\mathcal{B}_{m_t}$ que le courant ($\mathcal{B}_{m^*_{t-1}}$) tant que $\mathcal{B}_{m_t}$ n'a pas été élu modèle courant ou que $c^*_{m^*_{t-1}} = 0$ pour $\mathcal{B}_{m^*_{t-1}}$. Il ne pourra être élu modèle courrant que si on trouve un point de changement pour $\mathcal{B}_{m^*_{t-1}}$ et que la prochaine fenêtre (ou courante si $c^*_{m^*_{t-1}} = 0$) est fidèle à $\mathcal{B}_{m_t}$ sur les c_{min} premières observations. c_{min} contrôle donc la durée minimale d'un modèle en taille de fenêtre. c_{min} évite de sélectionner, après un point de changement, un autre modèle pour seulement quelques observations (sans signification statistique donc) avant de changer à nouveau de modèle (celui qui aurait dû être sélectionné). Cet effet est montré dans la section 4, Expériences et Résultats, lorsqu'aucune contrainte n'est mise sur c^*_m avec $\mathcal{B}_{m^*_t} \neq \mathcal{B}_{m^*_{t-1}}$. Dans un premier temps, $c_{min} = r$ et donc $c^*_m = r$ si $\mathcal{B}_{m_t} \neq \mathcal{B}_{m^*_{t-1}}$.

Si tous les modèles candidats ont un point de changement immédiat ou invalide ($c^*_{m_i} = 0, c^*_{m_j} < c_{min}$) alors un nouveau modèle est appris.

Autrement dit, nous pouvons continuer d'apprendre $\mathcal{B}_{m^*_{t-1}}$ jusqu'à trouver un point de changement, après quoi nous cherchons si un modèle (peut-être le même après mise à jour des paramètres et de la structure) le remplace ou si nous avons besoin d'en créer un nouveau, et ainsi de suite. C'est donc seulement à la suite d'un point de changement que nous cherchons parmi tous les modèles déjà connus.

Algorithme 1 : Main loop

Données : modèle précédent m^*_{t-1}, observations $\mathbf{w}[\tau - r, \tau]$, $\mathbf{w}[\tau, \tau + r]$
Données : $\mathbf{D} = \{\mathcal{B}_m\}$, $\mathcal{B}_{m^*_{t-1}}$

début
  $\Phi \leftarrow find\_match(\mathbf{D}, \mathbf{w})$
  si $\Phi \neq \{\}$ **alors**
    $m^*_t \leftarrow \arg\max_m \left\{ \frac{LL_{tr}(\mathbf{\Theta}, \mathcal{G})}{LL(\mathbf{w}:\mathbf{\Theta}, \mathcal{G})} : (LL, LL_{tr}, m) \in \Phi \right\}$
    si $m^*_t \neq m^*_{t-1}$ **alors**
      trouver la coupure c sur $\mathbf{w}[\tau - r, \tau + r]$
      mettre à jour le modèle précédent $\mathcal{B}_{m^*_{t-1}}$ sur $\mathbf{w}[\tau - r, \tau - r + c]$
    si $nouvelles\ observations \geq 2 * observations\ précédentes$ **alors**
      mettre à jour structure et paramètres de $\mathcal{B}_{m^*_t}$ sur $\mathbf{w}[\tau - r + c, \tau + r]$
      $observations\ précédentes \leftarrow$
       $observations\ précédentes + nouvelles\ observations$
    sinon
      mettre à jour paramètres de $\mathcal{B}_{m^*_t}$ sur $\mathbf{w}[\tau - r + c, \tau + r]$
    retourner
  trouver la coupure c sur $\mathbf{w}[\tau - r, \tau + r]$
  mettre à jour le modèle précédent $\mathcal{B}_{m^*_{t-1}}$ sur $\mathbf{w}[\tau - r, \tau - r + c]$
  apprendre nouveau modèle sur $\mathbf{w}[\tau - r + c, \tau + r]$

3.3. Apprentissage avec variables de domaines incompatibles

Comme le montre la Figure 2, le nombre de variables actives peut changer au cours du processus. Dans ce cas, on peut être amené à confronter un modèle et une base de données avec un nombre différent de variables. Si les variables de la base de données forment un sous-ensemble des variables du modèle, avec X_e dans $\mathcal{G}_m$ mais pas dans la base, nous réalisons des inférences pour estimer $P\left(X_i \mid \mathbf{U}_i \setminus X_e\right)$ et calculons ensuite les vraisemblances.

Si les variables du modèle forment un sous-ensemble des variables de la base, ces informations ne sont pas exploitables par le modèle et sont simplement mises de côté. Un tel modèle ne sera pas sélectionné pour la fenêtre courante. Si le domaine des variables Ω_{X_i} diffère, nous ajoutons les états manquants en utilisant les paramètres a priori (non informatifs) de Dirichlet α_{ijk} et calculons ensuite les vraisemblances.

Les algorithmes 1 et 2 décrivent notre cadre général pour l'apprentissage de processus non-stationnaires en temps réel par ns-DBNs. Bien que la prochaine section détaille nos expériences, il est intéressant de noter que la complexité de notre algorithme ne dépend pas de la taille de la base de données mais seulement de la taille de la fenêtre et du nombre de modèles connus, ce qui est une qualité importante pour l'apprentissage en ligne.

Algorithme 2 : Find match

Données : observations $\mathbf{w}[\tau, \tau + r]$
Données : $\mathbf{D} = \{\mathcal{B}_m\}$, $Dir(\{\alpha_{ijk}\})$-$(paramètres\ de\ Dirichlet)$

début
  $\Phi \leftarrow \{\}$
  pour $\mathcal{B}_m = (\mathbf{\Theta}, \mathcal{G})_m \in \mathbf{D}$ **faire**
    tant que $\exists X_e \in \mathcal{B}_m, X_e \notin \mathbf{w}$ **faire**
      $\forall j \in [\![1, c_e]\!]$, éliminer X_e par inférence :
      $P(X_i \mid (\mathbf{U}_i \setminus X_e) = j)$
    tant que $\exists X_e \in \mathbf{w}, X_e \notin \mathcal{B}_m$, **faire**
      retirer X_e
    tant que $\exists X_i \in \mathbf{w}, X_i = k$ **and** $X_i \in \mathcal{B}_m, k \notin \Omega_{X_i}$ **faire**
      $\Omega_{X_i} \leftarrow \Omega_{X_i} \cup k$
      $\theta_{ijk} \leftarrow \frac{\alpha_{ijk}}{N_{ij}+\alpha_{ij}}$
      $\theta_{ij\{o\neq k\}} \leftarrow \frac{N_{ijo}+\alpha_{ijo}}{N_{ij}+\alpha_{ij}}$
      pour $X_l \in \mathcal{B}_m : X_i \in \mathbf{U}_l$ **faire**
        $\forall j \in [\![1, c_i]\!]$, calculer par inférence :
        $P(X_l \mid (\mathbf{U}_l \setminus X_i) = j, X_i = k) \leftarrow P(X_l \mid (\mathbf{U}_l \setminus X_i) = j)$
    si $\frac{LL_{tr}(\mathbf{\Theta},\mathcal{G})}{LL(\mathbf{w}:\mathbf{\Theta},\mathcal{G})} \geq 0.97$ **alors**
      $\Phi \leftarrow \Phi \cup (LL, LL_{tr}, m)$
  $return\ \Phi$

4. Expériences et Résultats

Nos expériences consistent à modéliser des processus non-stationnaires simulés. En utilisant la librairie aGrUM (http://agrum.lip6.fr), nous avons généré un DBN de 10 nœuds par *time slice* de cardinalité moyenne 7 ($[\![3, 10]\!]$) et de degré moyen 3. Nous avons ensuite perturbé la structure et les paramètres du modèle en utilisant la distance de Hellinger (Beran, 1977) entre deux modèles comme critère d'arrêt. Plusieurs seuils ont été choisis pour observer à quel point deux réseaux doivent être distants pour être reconnus comme deux modèles distincts. Des distances de Hellinger supérieures à 0.8 impliquent toujours des changements de paramètres pour tous les nœuds et parfois des changements structuraux pour un petit sous-ensemble de nœuds. Des distances sous

ce seuil impliquent des changements de paramètres pour quelques nœuds, de faible degré, et un arc est parfois créé, ajoutant peu d'informations.

Les bases de données sont ensuite échantillonnées pour chaque modèle avant d'être combinées pour former une base de données unique de 600.000 événements. Différentes taille d'époques ont été utilisées pour étudier l'impact de la taille des échantillons vis-à-vis de la distance de Hellinger entre réseaux ainsi que différentes résolutions de la fenêtre glissante pour étudier le comportement de l'algorithme lorsque la fenêtre chevauche des données de deux modèles distincts (*i.e.* l'époque n'est pas un multiple de la résolution de la fenêtre). Nous avons lancé chaque jeu de test avec et sans fenêtre adaptative. Il est important de remarquer que notre algorithme n'a aucune information a priori quant au nombre de modèles, leurs variables et les domaines de ces dernières, le nombre de transitions ou leur périodicité.

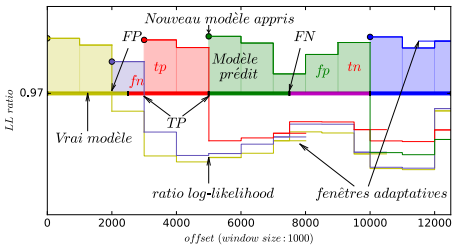

Figure 4. Lecture des figures.

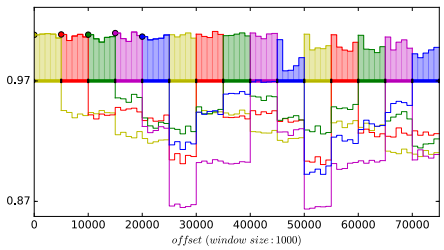

Figure 5. Résultats pour des époques de $5K$ observations, $Hellinger \approx 0.6$, fenêtre fixe

La figure 4 fictive explique comment lire les courbes et tables associées aux expériences, où FN signifie un faux négatif pour une transition (transitions manquées), FP signifie un faux positif pour une transition (transitions erronées) et TP signifie un vrai positif pour une transition (transitions correctes). Aussi, tp est le nombre de (vrai) événements appris par les modèles corrects, fp est le nombre de (faux) événements appris par de mauvais réseaux et fn est le nombre de (vrai) événements ratés par des modèles mais appris par d'autres. Les fenêtres adaptatives peuvent être vues lorsque les courbes sont étendues sur la gauche (pour le modèle courant qui déplace la fenêtre) ou sur la droite (modèles non sélectionnés qui ne déplacent pas la fenêtre). Dans les tables, l'erreur moyenne, minimale et maximale des points de changement est montrée, avec l'écart type, en terme de nombre d'événements. Finalement, la $précision$ $tp/(tp+fp)$ et le $rappel$ $tp/(tp+fn)$ sur les événements est calculée (voir Tables), c'est-à-dire la $précision$ moyenne et le $rappel$ moyen sur les modèles découverts. Le $rappel$ est le pourcentage d'événements corrects trouvés pour tous les événements corrects qui auraient dû être trouvés. La $précision$ est inversement proportionnelle au bruit (les événements générés depuis un autre modèle qui sont utilisés pour mettre à jour le modèle courant). Par souci d'espace, les résultats sont moyennés pour tous les seuils de distance Hellinger. Nous nous focalisons sur les cas où l'époque n'est pas un multiple de la résolution de la fenêtre et montrons un meilleur cas (Figure 5) et pire

cas (Figure 6, 7, 8, 9) avec et sans fenêtre adaptative. Les résultats pour les fenêtres statiques et dynamiques sont présentés dans les tables 1 et 2, respectivement.

4.1. Fenêtres statiques

La Figure 5 est un exemple d'une exécution réussie : l'époque est un multiple de la résolution de la fenêtre et par conséquent, la fenêtre glissante contient, à tout instant, uniquement des observations d'un modèle à la fois. Dans ces cas, les modèles et moments de transition sont correctement identifiés, avec ou sans fenêtre adaptative.

Cependant, des erreurs surviennent lorsque l'on utilise des résolutions arbitraires pour la fenêtre statique, comme le montrent les Figures 6 et 7.

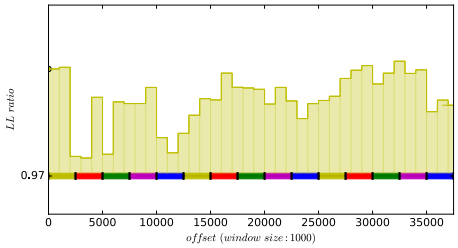

Figure 6. Résultats pour des époques de $2K5$ observations, $Hellinger \approx 0.6$, fenêtre fixe

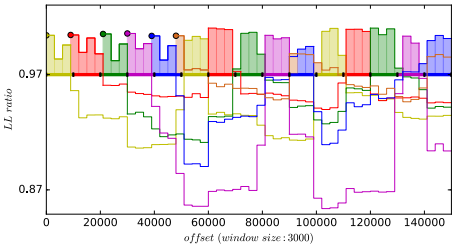

Figure 7. Résultats pour des époques de $10K$ observations, $Hellinger \approx 0.6$, fenêtre fixe

Dans le cas statique, comme montré en Table 1, deux phénomènes expliquent les faibles taux de $précision$ et $rappel$ obtenus pour certaines expériences. Le premier phénomène survient lorsque nous avons découvert moins de modèles que nous aurions dû, principalement avec des seuils sur la distance de Hellinger bas, auquel cas des transitions ont été ratées et certains modèles moyennent plusieurs modèles sous-jacents réels, augmentant leur bruit et rendant les transitions futures plus dures à détecter. Par conséquent, les FN de transitions augmentent alors que la $précision$ et le $rappel$ sur les événements diminuent. Un tel cas est exposé par la Figure 6 et les deux premières lignes de la Table 1, où la première ligne et la figure 6 montrent les résultats pour des modèles d'origine proches et la second ligne les résultats pour des modèles d'origine distants. Le second phénomène survient lorsque l'on a découvert davantage de modèles que nous aurions dû, principalement avec de grandes distances de Hellinger. Dans ce cas, la plupart des modèles en excès ont été créés lorsque la fenêtre chevauche des observations de deux modèles d'origine distincts, modélisant ainsi la transition elle-même (la prochaine fenêtre crée ou reconnaît le bon modèle), comme en Figure 7 (le modèle représenté en marron). Ainsi, nous avons deux transitions au lieu d'une, augmentant FP pour les transitions. La $précision$ et le $rappel$ sont moins affectés par ces FP puisque seules quelques transitions provoquent la création de modèles spécifiques, réduisant faiblement le $rappel$ d'autres réseaux découverts (à juste titre) mais augmentant sensiblement leur $précision$ (réduction du bruit).

Tableau 1. Résultats fenêtre fixe. Pour les coupures, erreur minimale, moyenne et maximale en nombre d'événements, avec écart-type.

époque	*taille fenêtre*	*FN*	*FP*	*TP*	*erreur moy.*	*écart type*	*min*	*max*	*précision*	*rappel*
2500	1000	1.0	0.0	0.0	NA	NA	NA	NA	0.2	1.0
2500	1000	0.0	0.0	1.0	251.046	249.998	0.0	500.0	0.829	0.875
2500	2000	0.602	0.0	1.0	521.052	361.644	0.0	1000.0	0.5	0.952
5000	1500	0.101	0.035	0.965	351.216	258.546	0.0	1000.0	0.833	0.935
5000	3000	0.0	0.06	0.94	752.1	579.236	0.0	2000.0	0.856	0.854
10000	1500	0.0	0.131	0.869	381.356	295.694	0.0	1000.0	0.961	0.96
10000	3000	0.1017	0.0083	0.992	772.81	601.843	0.0	2000.0	0.833	0.929
15000	2000	0.0	0.204	0.795	512.82	499.835	0.0	1000.0	0.963	0.958

Les résultats pour les fenêtres statiques pourraient être pires : dans nos expériences, une époque n'est pas un multiple de la résolution de la fenêtre mais un multiple de l'époque peut être un multiple de la résolution de la fenêtre, dans quel cas la fenêtre commence ou termine au niveau d'une vraie transition, ce qui a pour effet "d'augmenter" potentiellement la probabilité de correctement identifier une transition et/ou un modèle. Ainsi, FN et l'erreur sur les points de changement pourraient être plus grande alors que la $précision$ et le $rappel$ pourraient être plus faibles.

4.2. Fenêtres adaptatives

Tableau 2. Résultats fenêtre adaptative, avec les colonnes comme en table 1.

époque	*taille fenêtre*	*FN*	*FP*	*TP*	*erreur moy.*	*écart type*	*min*	*max*	*précision*	*rappel*
2500	1000	0.0	0.0	1.0	4.399	18.045	0.0	254.0	0.998	0.998
2500	2000	0.0	0.0	1.0	2.435	4.683	0.0	38.5	0.999	0.999
5000	1500	0.0	0.0	1.0	3.0966	7.784	0.0	62.5	0.999	0.999
5000	3000	0.0	0.0	1.0	8.702	43.383	0.0	390.5	0.998	0.998
10000	1500	0.0	0.0	1.0	32.923	80.526	0.0	311.5	0.997	0.996
10000	3000	0.0	0.0	1.0	19.559	103.199	0.0	778.0	0.998	0.998
15000	2000	0.0	0.0	1.0	8.551	32.423	0.0	202.5	0.999	0.999

Les résultats pour les fenêtres adaptatives, montrés en Table 2, révèlent que la taille de la fenêtre a peu d'impact sur l'identification correcte des transitions et des modèles, et ceci devrait être vrai tant que la résolution de la fenêtre est plus petite que celle d'une époque. Étonnamment, les résultats ne sont pas plus mauvais pour de petites époques étant donné la taille des réseaux. Avec les fenêtres adaptatives, les deux phénomènes précédents sont résolus en cherchant le point de changement optimal, comme en Figures 8 et 9 : dans le premier cas, nous n'apprenons pas depuis des fenêtres qui chevauchent deux modèles ce qui réduit le bruit et évite d'accroître la difficulté d'identification des transitions futures. Dans le second cas, chercher un point de changement évite en soi-même la création d'un modèle représentant la transition seule.

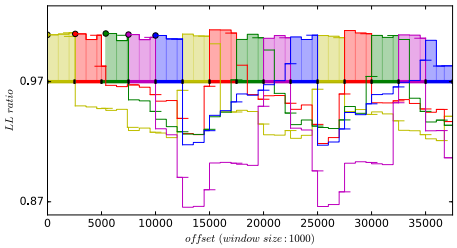

Figure 8. Résultats pour des époques de $2K5$ observations, $Hellinger \approx 0.6$, fenêtre adaptative

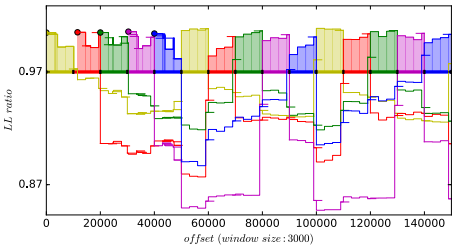

Figure 9. Résultats pour des époques de $10K$ observations, $Hellinger \approx 0.6$, fenêtre adaptative

La capacité de l'algorithme à ajouter des modalités à des variables connues évite la création de modèles redondants dans les deux cas, ce qui réduit FP pour les transitions et est d'une importance cruciale pour les *outliers* qui surviennent de temps à autre et doivent être incorporés aux modèles déjà existants.

4.3. Résultats depuis Équation (11)

Dans cette section nous présentons les résultats pour le même jeu d'expériences que précédemment mais en utilisant directement l'équation (11) sans passer par (9). Seuls les résultats pour les fenêtres adaptatives sont donc présentés étant donné que l'équation (11) est l'adaptation de la fenêtre.

Tableau 3. Résultats fenêtre adaptative, toutes distances de hellinger confondues.

époque	*taille fenêtre*	*FN*	*FP*	*TP*	*erreur moy.*	*écart type*	*min*	*max*	*précision*	*rappel*
2500	1000	0.0	0.0	1.0	2.58	9.69	0	103.5	0.999	0.999
2500	2000	0.0	0.0	1.0	1.65	2.78	0	25	0.999	0.999
5000	1500	0.0	0.0	1.0	1.97	3.47	0	23.5	0.999	0.999
5000	3000	0.0	0.0	1.0	75.23	155.166	0	406.5	0.987	0.985
10000	1500	0.0	0.0	1.0	6.059	19.158	0	102	0.999	0.999
10000	3000	0.0	0.0	1.0	2.10	3.21	0	13.5	0.999	0.999
15000	2000	0.0	0.0	1.0	24.38	43.969	0	163	0.998	0.998

Les résultats montrés en Table 3 sont aussi bons ou meilleurs que les résultats en Table 2, notamment sur la précision des points de changement. En effet, lorsqu'on utilise l'équation (9), il se peut qu'une partie de la fenêtre (en fin de fenêtre) n'ait pas une vraisemblance suffisamment basse pour faire échouer le test statistique. La fenêtre est alors apprise, introduisant du bruit et augmentant le risque de ne pas identifier la transition courante ou de l'identifier plus tardivement (ou d'avoir un point de changement plus lointain qu'il ne faut, à cause du changement de paramètre et/ou structure induit par la fenêtre précédente).

Utiliser directement l'équation (11) sans passer par (9) permet aussi de s'affranchir du sampling et offre donc un gain de temps significatif (pour chaque création de nouveau modèle et ré-estimation de la structure).

Les tables 4, 5 présentent les résultats de la table 3 selon les distances de Hellinger qui séparent les modèles d'origine. On remarque que les résultats de la table 3 sont principalement guidés par ceux de la table 4, c'est-à-dire par les expériences avec des modèles plus facilement confondables. Lorsque les modèles sont distants, la plus grande imprécision sur le temps d'un point de changement est 7. Ces résultats sont attendus puisqu'il existe un seuil en dessous duquel il n'est plus possible de différencier deux modèles. Plus les modèles sont confondables, plus l'imprécision des points de changement augmente. Nous verrons dans la prochaine section que la modification d'un paramètre permet de contrôler la finesse de la détection de ces points. On peut donc trouver la valeur du paramètre qui nous permet de descendre jusqu'à la distance de Hellinger souhaitée.

Tableau 4. Résultats fenêtre adaptative, $Hellinger \approx 0.6$.

époque	*taille fenêtre*	*FN*	*FP*	*TP*	*erreur moy.*	*écart type*	*min*	*max*	*précision*	*rappel*
2500	1000	0	0	1	4.83	18.51	0	200	0.998	0.998
2500	2000	0	0	1	2.975	4.73	0	3	0.999	0.999
5000	1500	0	0	1	3.63	6.18	0	43	0.999	0.999
5000	3000	0	0	1	150.202	309.61	0	809	0.974	0.970
10000	1500	0	0	1	11.644	37.32	0	200	0.999	0.999
10000	3000	0	0	1	3.78	5.47	0	23	0.999	0.999
15000	2000	0	0	1	48.513	87.31	0	323	0.997	0.997

Tableau 5. Résultats fenêtre adaptative, $Hellinger \geq 0.8$.

époque	*taille fenêtre*	*FN*	*FP*	*TP*	*erreur moy.*	*écart type*	*min*	*max*	*précision*	*rappel*
2500	1000	0	0	1	0.339	0.867	0	7	0.999	0.999
2500	2000	0	0	1	0.37	0.83	0	7	0.999	0.999
5000	1500	0	0	1	0.30	0.76	0	4	0.999	0.999
5000	3000	0	0	1	0.27	0.72	0	4	0.999	0.999
10000	1500	0	0	1	0.47	0.997	0	4	0.999	0.999
10000	3000	0	0	1	0.42	0.94	0	4	0.999	0.999
15000	2000	0	0	1	0.256	0.629	0	3	0.999	0.999

4.4. Influence des paramètres

Dans cette section nous étudions l'impact des paramètres de l'algorithme sur la qualité des modèles découverts. Les trois paramètres d'intérêt sont :

– k, Équation (10), le nombre de fenêtre passées gardées en mémoire pour l'algorithme de point de changement.

– α, Équation (10), le facteur de vraisemblance qui sert de pénalité dans $f(c)$ et contrôle quand un point est un potentiel point de changement.

– c_{min}, Équation (11), qui correspond à la durée minimale d'un mode et donc au nombre de points minimum (de la nouvelle fenêtre) pour le test statistique de présence de point de changement. Il permet de ne pas choisir de modèles candidats fallacieux.

Les résultats sont séparés selon la distance de hellinger inter-modèles (ceux utilisés pour générer les base de données). Pour chaque point nous avons moyenné les résultats de 7 expériences (celles présentées dans les Tables précédentes) donc avec des époques et fenêtres de tailles différentes les unes des autres.

Un seul paramètre varie à la fois, les autres sont maintenus fixes. k, α et c_{min} ont comme valeur par défaut (valeur utilisée pour toutes les expériences précédentes), 4, 0.005 et 100, respectivement.

4.4.1. Étude de k

Nous faisons varier k, le nombre de fenêtres passées retenues pour détecter un prochain point de changement, avec $c_{min} = 100$ et $\alpha = 0.005$. Le point le plus à droite ($k = 40$) sur les Figures 10, 11, 12, 13, 14 et 15 correspond à une asymptote (i.e. toutes les fenêtres précédentes sont retenues) et à été fixé à 40 pour ne pas écraser les courbes sur la gauche et parce-qu'il y a au moins 40 fenêtres de données pour chacun des modèles d'origine (120000 évènements par modèle et des fenêtres de taille maximale 3000).

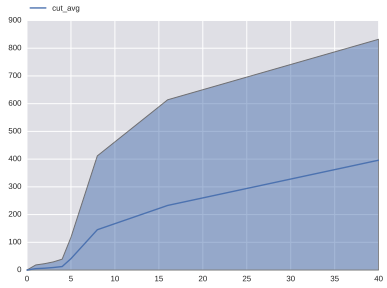

Figure 10. Erreur moyenne en fonction de k, $Hellinger \approx 0.6$

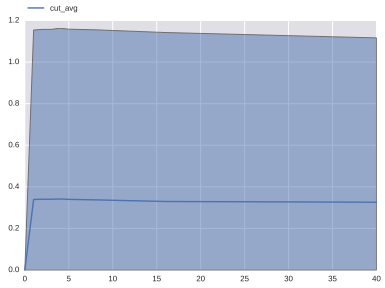

Figure 11. Erreur moyenne en fonction de k, $Hellinger \geq 0.8$

Le point $(0, 0)$ a été fixé par convention puisqu'il est nécessaire d'avoir des données précédant les nouvelles observations pour trouver un point de changement (vis-à-vis du modèle) dans ces dernières. Avec $k = 0$, aucun point de changement n'est trouvé et nous n'avons par conséquent pas de valeur pour l'erreur sur les points de changement.

Intuitivement, plus le nombre de fenêtres retenues augmente plus l'imprécision sur les points de changements augmente (Figure 10). En effet, plus l'apprentissage d'un modèle se renforce avec le temps pour converger plus la vraisemblance de données

issues de son modèle d'origine augmente. Ainsi, retenir des fenêtres de données trop lointaines augmente l'écart-type de la loi à gauche de tout point de changement potentiel ce qui a pour effet de rendre les détections de changements doux difficiles. Les changements plus brusques, figure 11, sont plus facilement détectables.

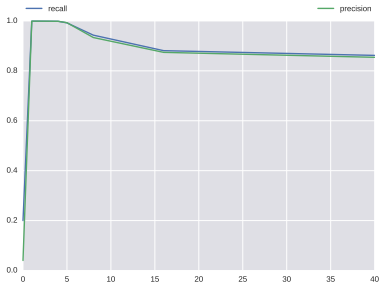

Figure 12. Rappel et precision en fonction de k, $Hellinger \approx 0.6$

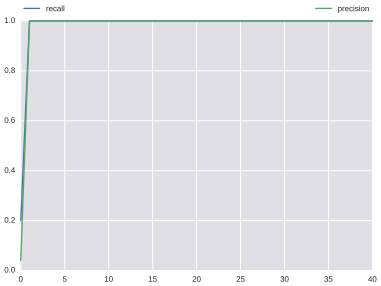

Figure 13. Rappel et precision en fonction de k, $Hellinger \geq 0.8$

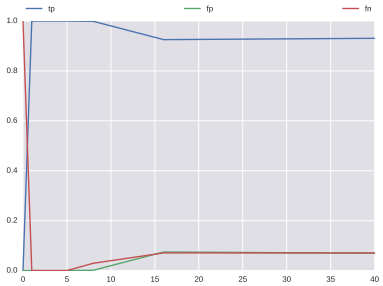

Figure 14. TP, FP, FN en fonction de k, $Hellinger \approx 0.6$)

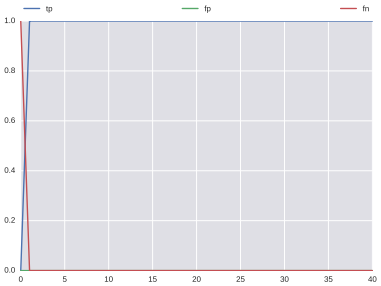

Figure 15. TP, FP, FN en fonction de k, $Hellinger \geq 0.8$

L'augmentation de k introduit donc d'abord une augmentation de l'imprécision sur les points de changements (Figure 10) ce qui se traduit par une augmentation du bruit dans les réseaux appris et donc une diminution du rappel sur les évènements (Figure 12). Comme l'augmentation du bruit pour un réseau donné signifie une diminution de la précision d'un autre réseau (son taux de faux négatifs sur les évènements augmente puisque ces derniers ont été absorbés par un autre modèle), la précision moyenne diminue de concert. Le rappel et la précision restent cependant corrects, malgré l'augmentation avec $k \geq 6$ du taux de faux négatifs et faux positifs pour les transitions (Figure 14). Le rappel et la précision chutent alors davantage puisque toute transition ratée ou erronée introduit du bruit pour le modèle courant et une perte de précision pour le modèle qui aurait dû être sélectionné.

Le cas avec des modèles originaux très distants est insensible à l'augmentation de k (Figures 11, 13, 15). L'erreur sur les points de changement a une moyenne inférieure à 1 évènement et l'écart-type est très faible (Figure 11).

Globalement k n'est pas le paramètres qui a le plus d'influence sur notre algorithme et les performances se maintiennent relativement bien ; le rappel et la précision restant élevés, les capacités de prédiction des modèles appris restent correctes. Les meilleurs résultats sont atteints avec $k \in \{1, 2, 3\}$.

4.4.2. Étude de α

Nous faisons varier α, le facteur de vraisemblance qui sert de pénalité dans $f(c)$ et contrôle quand un point est un potentiel point de changement, en maintenant $c_{min} = 100$ et $k = 4$.

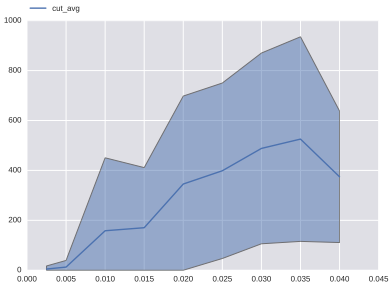

Figure 16. Erreur moyenne en fonction de α, $Hellinger \approx 0.6$

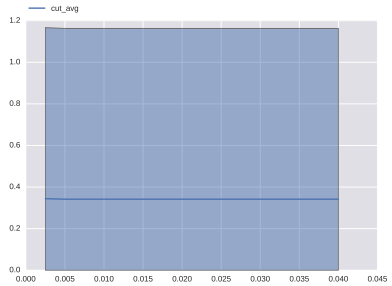

Figure 17. Erreur moyenne en fonction de α, $Hellinger \geq 0.8$

Plus intuitivement que précédemment, puisque α contrôle directement de combien un point de changement doit améliorer la vraisemblance de la fenêtre pour un modèle considéré. L'augmentation de α diminue le facteur d'amélioration attendu (puisque la log-vraisemblance est négative). α contrôle donc indirectement la distance minimale détectable et par conséquent entre modèles appris. Un α trop grand pour un processus qui évolue doucement d'un mode à un autre entraîne une augmentation de l'erreur sur les points de coupures (Figure 16).

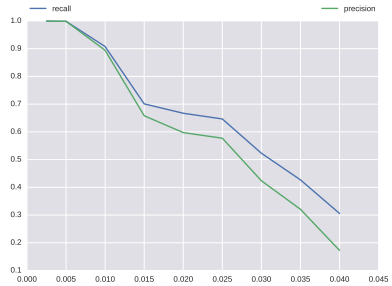

Figure 18. Rappel et precision en fonction de α, $Hellinger \approx 0.6$

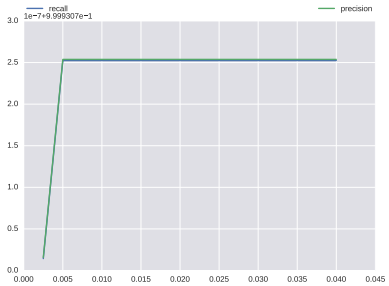

Figure 19. Rappel et precision en fonction de α, $Hellinger \geq 0.8$

Entre $\alpha = 0.02$ et $\alpha = 0.025$, pour des distances de Hellinger d'environ 0.6, certains jeux d'expériences commencent à échouer complètement (aucune transition

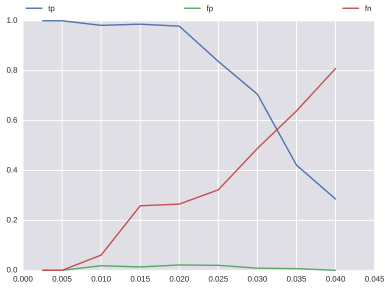

Figure 20. TP, FP, FN en fonction de α, $Hellinger \approx 0.6$

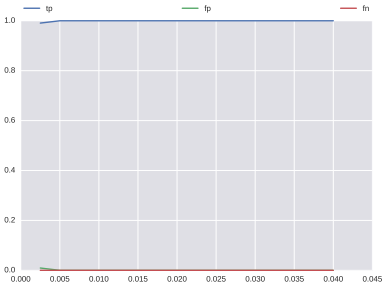

Figure 21. TP, FP, FN en fonction de α, $Hellinger \geq 0.8$

n'est détectée). Sur la Figure 20 cela se traduit par $TP + FP < 1$ (l'ensemble des transitions découvertes doit sommer à 1 pour chaque expérience autrement nous avons manqué toutes les transitions ($FN = 1$) pour cette expérience). À partir de cette valeur de α, la plupart des jeux d'expériences perdent la capacité de détecter des points de coupures petit-à-petit pour certains modèles ; nous avons une augmentation du taux de faux négatifs (Figure 20) pour des modèles entiers qui sont confondus avec un autre. D'autre modèles au sein de la même expérience sont correctement détectés mais présentent beaucoup de bruit à cause de l'erreur élevée des points de changements. Toutes expériences confondues le rappel et la précision diminuent donc drastiquement avec un α destiné à détecter des changements trop brusques (Figure 18) pour un processus qui évolue doucement d'un mode à un autre.

Il est intéressant de noter que α ne fait pas monter le taux de faux positifs pour les transitions mais augmente les faux négatifs (Figure 20). Dans certaines applications (notamment pour les IDS), il est préférable d'avoir moins d'alertes, mais pertinentes, que beaucoup d'alertes dont la plupart sont de fausses alarmes.

Encore une fois, les processus aux changements brusques sont peu sensibles à la valeur de α (Figures 17, 19, 21).

α est le paramètre qui influence le plus notre algorithme en imposant une limite sur la distance qui sépare les modèles découverts. Pour des modèles très proches, une valeur de α légèrement supérieure au "paramètre idéal" (qui permettrai de détecter exactement les modèles d'origine) pourrait éviter la création de modèles redondants car trop proches (pour des distances de Hellinger plus petites que 0.1 par exemple).

Pour détecter des modèles séparés par une distance de Hellinger plus grande que 0.6, $\alpha = 0.005$ semble idéal.

4.4.3. Étude de c_{min}

Finalement, nous faisons varier c_{min}, la durée minimale d'un mode (ou le nombre minimal de points autorisé d'une nouvelle fenêtre après un point de coupure pour un modèle $m_t^* \neq m_{t-1}^*$).

c_{min} fait office de taille d'échantillon pour le test en équation 10, 11. Intuitivement, que dire d'un transition d'un modèle vers un autre pour une durée de 2 points avant de transiter à nouveau vers un autre modèle ? Si la vraisemblance moyenne d'un modèle m_i est meilleure que la vraisemblance moyenne d'un modèle m_j avec $c_i^* \lll c_j^*$, on peut se poser la question du meilleur modèle à choisir. Il est en effet possible d'avoir plusieurs *matchs* sur une même fenêtre adaptée différemment par chaque modèle candidat.

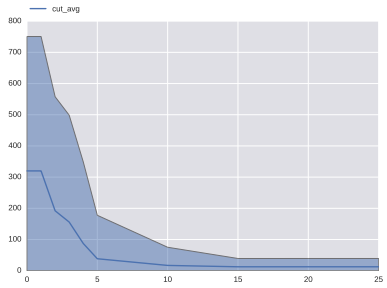

Figure 22. Erreur moyenne en fonction de c_{min}, $Hellinger \approx 0.6$

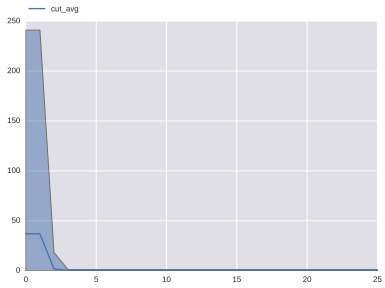

Figure 23. Erreur moyenne en fonction de c_{min}, $Hellinger \geq 0.8$

En règle générale, plus la taille d'échantillon est importante plus les tests statistiques sont fiables. L'augmentation de c_{min} diminue donc l'erreur sur les points de changement que ce soit pour les expériences avec des modèles relativement proches (Figure 22) ou distants (Figure 23).

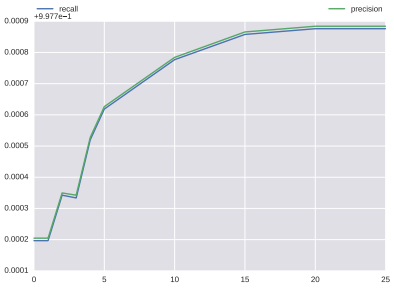

Figure 24. Rappel et precision en fonction de c_{min}, $Hellinger \approx 0.6$

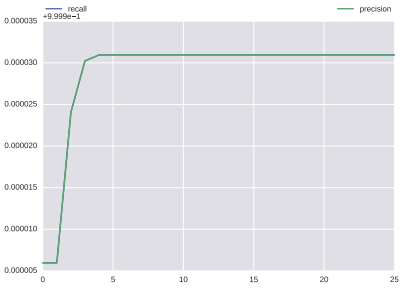

Figure 25. Rappel et precision en fonction de c_{min}, $Hellinger \geq 0.8$

L'augmentation de c_{cmin} permet d'éviter les transitions archaïques soit le taux de faux positifs (Figure 26, 27). Conjointement avec une meilleure précision des points de changements, le rappel et la précision augmentent (Figures 24, 25).

Pour les distances utilisés une valeur de $c_{min} \geq 20$ semble suffisante. Cependant, il est intéressant de remarquer que plus les modèles sont proches, plus c_{min} a besoin d'être augmenté avant d'atteindre son asymptote en terme de rappel et précision (Figures 24, 25). Avec des modèles séparés par une distance de Hellinger plus petite que

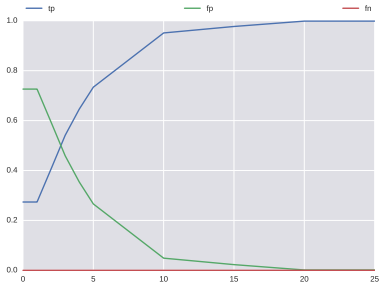

Figure 26. TP, FP, FN en fonction de c_{min}, $Hellinger \approx 0.6$

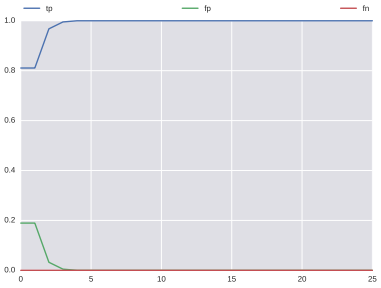

Figure 27. TP, FP, FN en fonction de c_{min}, $Hellinger \geq 0.8$

0.6, il faudrait sans doute fixer c_{cmin} aux alentours de 100. En effet, pour distinguer deux distributions proches, il faut beaucoup plus de données.

4.5. Première application sur un cas réel

Nous présentons les premiers résultats obtenus sur des logs apache (une ligne est présentée ci-dessous).

```
173.36.240.171 - - [01/Jun/2015:00:01:19 +0000] "GET
/projects/scapy/ HTTP/1.1" 200 6205
"http://r.search.yahoo.com/_ylt=AwrTccnLoGtVzh8A_NUnnIlQ;
_ylu=X3oDMTByYnR1Zmd1BGNvbG8DZ3ExBHBvcwMyBHZ0aWQDBHNlYwNzcg
--/RV=2/RE=1433145676/RO=10/RU=http%3a%2f%2fsecdev.org%2f
projects%2fscapy%2f/RK=0/RS=DV3KSkU3cpgfOsdls5DOVKn1zrs-
" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:38.0)
Gecko/20100101 Firefox/38.0"
```

Le serveur a enregistré un mois de logs, présentant 29341 adresses ip et quelques anomalies / patterns d'attaque (5) on été identifiés au préalable par des experts (manuellement et par signature).

Sur ce mois, 33 modèles ont été découverts. Pour chaque ip, nous avons un ensemble de modèles ; certains sont partagés par plusieurs adresses ip et d'autres sont uniques. Un modèle apparaît seulement pendant les attaques et inclut toutes les occurrences d'attaques connues détectables.

Nous avons donc identifié un modèle qui a capturé les dynamiques anormales qui surviennent lors des attaques. Cependant, toutes les attaques n'ont pas été détectées car toutes ne sont pas détectables telles quelles. En effet, plusieurs adresses ip n'apparaissent qu'une seule fois et ne réalisent qu'une action ; nous n'avons donc pas de trajectoire pour ces dernières et il y a peu de choses à faire de ces données trop parcellaires.

5. Conclusions et Travaux Futurs

Nous avons construit un cadre général utilisant les réseaux bayésiens dynamiques pour apprendre des processus non-stationnaires de manière continue, dirigé comme un compromis entre rapidité et précision. Les expériences montrent de très bon résultats sur les premiers jeux de données simulées et réelles. Il serait intéressant (et nécessaire) d'expérimenter davantage avec α pour descendre à de plus basses distances entre comportements. Plusieurs améliorations viennent à l'esprit : nous avons mentionné un seuil dynamique sur les ratios de log-vraisemblance pour prendre en compte la convergence d'un modèle, ainsi que le besoin d'une procédure de fusion et suppression entre modèles, puisque nous nous attendons à ce que les résultats soient de plus en plus mauvais au fur et à mesure que la distance entre réseaux d'origine diminue. Bien qu'une procédure de suppression naïve pourrait consister à utiliser un paramètre pour chaque modèle, décroissant avec le temps lorsque le modèle n'est pas rencontré, la fusion de modèles nécessite de comparer leurs probabilités jointes, ce qui implique des calculs lourds. Le problème du point de changement optimal peut aussi être poussé plus loin. Une amélioration d'importance serait de modéliser les transitions entre modèles et prédire dans une certaine mesure le prochain comportement attendu ou identifier les signaux faibles qui permettraient de telles prédictions. Finalement, nous appliquerons cet algorithme dans le cadre de la détection d'anomalies dans un Système de Détection d'Intrusion (hôte et réseau).

Remerciements

Ce travail est supporté par Akheros S.A.S./bourse ANRT CIFRE #2014/0268 et le projet européen SCISSOR H2020-ICT-2014-1 #644425.

Bibliographie

Akaike H. (1998). Information theory and an extension of the maximum likelihood principle. In *Selected papers of hirotugu akaike*, p. 199–213. Springer.

An X., Jutla D., Cercone N. (2006). Privacy intrusion detection using dynamic bayesian networks. In *Acm international conference proceeding series*, vol. 156, p. 208–215.

Beran R. (1977). Minimum hellinger distance estimates for parametric models. *The Annals of Statistics*, p. 445–463.

Casella G., George E. I. (1992). Explaining the gibbs sampler. *The American Statistician*, vol. 46, n° 3, p. 167–174.

Charitos T., Van Der Gaag L. C., Visscher S., Schurink K. A., Lucas P. J. (2009). A dynamic bayesian network for diagnosing ventilator-associated pneumonia in icu patients. *Expert Systems with Applications*, vol. 36, n° 2, p. 1249–1258.

Chickering D., Geiger D., Heckerman D. (1995). Learning bayesian networks: Search methods and experimental results. In *proceedings of fifth conference on artificial intelligence and statistics*, p. 112–128.

Dean T., Kanazawa K. (1989). A model for reasoning about persistence and causation. *Computational intelligence*, vol. 5, n° 2, p. 142–150.

Debar H., Dacier M., Wespi A. (1999). Towards a taxonomy of intrusion-detection systems. *Computer Networks*, vol. 31, n° 8, p. 805–822.

Gonzales C., Dubuisson S., Manfredotti C. (2015). A new algorithm for learning non-stationary dynamic bayesian networks with application to event detection. In *The twenty-eighth international flairs conference.*

Grzegorczyk M., Husmeier D. (2009). Non-stationary continuous dynamic bayesian networks. In *Advances in neural information processing systems*, p. 682–690.

Grzegorczyk M., Husmeier D. (2011). Non-homogeneous dynamic bayesian networks for continuous data. *Machine Learning*, vol. 83, n° 3, p. 355–419.

Grzegorczyk M., Husmeier D., Edwards K. D., Ghazal P., Millar A. J. (2008). Modelling non-stationary gene regulatory processes with a non-homogeneous bayesian network and the allocation sampler. *Bioinformatics*, vol. 24, n° 18, p. 2071–2078.

Heckerman D., Geiger D., Chickering D. M. (1995). Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, vol. 20, n° 3, p. 197–243.

Kruegel C., Mutz D., Robertson W., Valeur F. (2003). Bayesian event classification for intrusion detection. In *Computer security applications conference, 2003. proceedings. 19th annual*, p. 14–23.

Kullback S., Leibler R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, vol. 22, n° 1, p. 79–86.

Lerner U., Parr R., Koller D., Biswas G. *et al.* (2000). Bayesian fault detection and diagnosis in dynamic systems. In *Aaai/iaai*, p. 531–537.

Mitra V., Nam H., Espy-Wilson C. Y., Saltzman E., Goldstein L. (2011). Gesture-based dynamic bayesian network for noise robust speech recognition. In *Acoustics, speech and signal processing (icassp), 2011 ieee international conference on*, p. 5172–5175.

Murphy K. P. (2002). *Dynamic bayesian networks: representation, inference and learning*. Thèse de doctorat non publiée, University of California, Berkeley.

Mutz D., Valeur F., Vigna G., Kruegel C. (2006). Anomalous system call detection. *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, n° 1, p. 61–93.

Ourston D., Matzner S., Stump W., Hopkins B. (2003). Applications of hidden markov models to detecting multi-stage network attacks. In *System sciences, 2003. proceedings of the 36th annual hawaii international conference on*, p. 10–pp.

Pearl J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Francisco, CA, USA, Morgan Kaufmann Publishers Inc.

Perron P. (1989). The great crash, the oil price shock, and the unit root hypothesis. *Econometrica: Journal of the Econometric Society*, p. 1361–1401.

Rissanen J. (1978). Modeling by shortest data description. *Automatica*, vol. 14, n° 5, p. 465–471.

Robinson J. W., Hartemink A. J. (2009). Non-stationary dynamic bayesian networks. In *Advances in neural information processing systems*, p. 1369–1376.

Robinson J. W., Hartemink A. J. (2010). Learning non-stationary dynamic bayesian networks. *The Journal of Machine Learning Research*, vol. 11, p. 3647–3680.

Schwarz G. (1978, 03). Estimating the dimension of a model. *Ann. Statist.*, vol. 6, n° 2, p. 461–464. Consulté sur http://dx.doi.org/10.1214/aos/1176344136

Sicard M., Baudrit C., Leclerc-Perlat M., Wuillemin P.-H., Perrot N. (2011). Expert knowledge integration to model complex food processes. application on the camembert cheese ripening process. *Expert Systems with Applications*, vol. 38, n° 9, p. 11804–11812.

Song L., Kolar M., Xing E. P. (2009). Time-varying dynamic bayesian networks. In *Advances in neural information processing systems*, p. 1732–1740.

Xu J., Shelton C. R. (2008). Continuous time bayesian networks for host level network intrusion detection. In *Machine learning and knowledge discovery in databases*, p. 613–627. Springer.

Xu J., Shelton C. R. (2010). Intrusion detection using continuous time bayesian networks. *Journal of Artificial Intelligence Research*, p. 745–774.

Yeung D.-Y., Ding Y. (2003). Host-based intrusion detection using dynamic and static behavioral models. *Pattern recognition*, vol. 36, n° 1, p. 229–243.

Zanero S., Serazzi G. (2008). Unsupervised learning algorithms for intrusion detection. In *Network operations and management symposium, 2008. noms 2008. ieee*, p. 1043–1048.

Zivot E., Andrews D. W. (2012). Further evidence on the great crash, the oil-price shock, and the unit-root hypothesis. *Journal of Business & Economic Statistics*.