Table des matières

Remerciements **1**

Abréviations **3**

Liste des publications de l'auteur dans le cadre de la thèse **5**

Introduction **7**
1 Les origines des télécommunications et des objets connectés 7
1.1 Des origines du besoin de télécommuniquer 7
1.2 Les premiers systèmes de télécommunication 7
1.3 Les réseaux de télécommunication modernes 8
1.4 Le développement de la téléphonie mobile 9
1.5 L'arrivée des objets connectés et la 5G 9
1.6 Bilan 10
2 De l'imprévisibilité du protocole d'une couche sachant ses couches inférieures . 10
2.1 Constitution d'un échantillon de protocoles 11
2.2 Prévisibilité du protocole de la couche liaison de données sachant celui de la couche physique 13
3 Problématique, famille d'adaptabilité choisie et état de l'art 15
3.1 Problématique 15
3.2 Choix du type d'adaptabilité 16
3.3 État de l'art des systèmes d'établissement automatique de communication avec un protocole inconnu 19
4 Démarche, orientation et organisation de la thèse 20
4.1 Démarche 20
4.2 Configuration d'étude de la thèse 21
4.3 Solution proposée 21
4.4 Plan du manuscrit 22

1 Étude des algorithmes VDV, AC et LDA **25**
1.1 Objectifs 26
1.2 Algorithme Variance of the Distribution of Variances (VDV) 26
1.2.1 Principe 26
1.2.2 Algorithme 27
1.2.3 Application à la configuration d'étude de la thèse 27
1.3 Algorithme d'Aho-Corasick (AC) 30
1.3.1 Principe 30

1.3.2 Algorithme . . . 31
1.3.3 Application à la configuration d'étude de la thèse . . . 31
1.4 Algorithme Latent Dirichlet Allocation (LDA) . . . 33
1.4.1 Choix de la loi de Dirichlet . . . 34
1.4.2 Modèle génératif . . . 34
1.4.3 Inférence de Θ et Φ . . . 37
1.4.4 Échantillonnage de Gibbs . . . 39
1.4.5 Perplexité . . . 40
1.4.6 Algorithme . . . 40
1.4.7 Application à la configuration d'étude de la thèse . . . 42
1.5 Conclusion . . . 43

2 Simulations comparatives des algorithmes VDV, AC et LDA 45
2.1 Modèle de simulation . . . 45
2.1.1 Génération de la trace liaison de données . . . 45
2.1.2 Analyse de la trace liaison de données . . . 47
2.1.3 Calcul des performances et comparaison . . . 47
2.2 Déroulement des simulations . . . 50
2.3 Simulations de l'algorithme VDV . . . 51
2.3.1 Paramètres . . . 51
2.3.2 Influence de la longueur maximale des séquences recherchées . . . 51
2.3.3 Influence du nombre de flux générés . . . 53
2.3.4 Influence du seuil de filtrage des séquences . . . 54
2.4 Simulations de l'algorithme AC . . . 57
2.4.1 Paramètres . . . 57
2.4.2 Influence de la longueur maximale des séquences recherchées . . . 57
2.4.3 Influence du seuil de filtrage des séquences . . . 59
2.4.4 Influence du seuil de fusion des séquences . . . 60
2.5 Simulations de l'algorithme LDA . . . 62
2.5.1 Paramètres . . . 62
2.5.2 Influence de la longueur maximale des séquences recherchées . . . 63
2.5.3 Influence du nombre de mots-clés recherchés . . . 66
2.5.4 Influence du gradient maximal de perplexité avant arrêt de l'échantillonneur de Gibbs . . . 67
2.5.5 Influence du gradient maximal de probabilité pour la sélection des n-grams 69
2.5.6 Influence du paramètre 'alpha' de la distribution des mots-clés . . . 71
2.5.7 Influence du paramètre 'bêta' de la distribution des n-grams . . . 73
2.6 Comparaison des performances des algorithmes et conclusion . . . 73
2.6.1 Paramètres des simulations . . . 74
2.6.2 Discussion des résultats de la comparaison . . . 74
2.7 Conclusion . . . 78

3 Proposition d'un nouveau modèle : BaNet3F 81
3.1 Réseaux Bayésiens . . . 81
3.1.1 Principes fondamentaux et représentation des réseaux Bayésiens . . . 81
3.1.2 Inférence exacte dans un réseau Bayésien . . . 84
3.1.3 Inférence de Viterbi dans un réseau Bayésien . . . 92

3.1.4 Apprentissage de paramètres dans les réseaux Bayésiens avec l'algorithme *Expectation-Maximization* (EM) . . . 97
3.2 Réseaux Bayésiens dynamiques . . . 98
3.2.1 Principes fondamentaux et représentation des réseaux Bayésiens dynamiques . . . 98
3.2.2 Inférence exacte dans un réseau Bayésien dynamique . . . 99
3.2.3 Inférence de Viterbi dans un réseau Bayésien dynamique . . . 104
3.2.4 Apprentissage dans les réseaux Bayésiens dynamiques avec l'algorithme *Expectation-Maximization* (EM) . . . 106
3.3 Modèle proposé : BaNet3F . . . 108
3.3.1 Principe fondamental et vue générale du modèle . . . 108
3.3.2 Réseau Bayésien employé . . . 112
3.3.3 Apprentissage des paramètres à l'aide de l'EM . . . 115
3.3.4 Détermination du découpage le plus probable à l'aide de l'algorithme AMBV . . . 118
3.3.5 Validation du découpage à l'aide de sa probabilité . . . 125
3.4 Conclusion . . . 126

4 Performances du modèle BaNet3F sur traces générées et traces réelles 129
4.1 Paramètres et métriques . . . 130
4.1.1 Paramètres . . . 130
4.1.2 Métriques . . . 131
4.2 Performances obtenues avec les traces générées . . . 132
4.2.1 Générateur de traces . . . 132
4.2.2 Cadre des simulations . . . 134
4.2.3 Influence du nombre maximal de champs par trame sur les performances de BaNet3F . . . 137
4.2.4 Influence du seuil relatif de validation du découpage sur les performances de BaNet3F . . . 144
4.2.5 Influence du nombre maximal d'unités élémentaires considérées simultanément sur les performances de BaNet3F . . . 147
4.2.6 Bilan . . . 152
4.3 Performances obtenues avec les traces réelles . . . 154
4.3.1 Caractéristiques de la trace et cadre des simulations . . . 154
4.3.2 Influence du nombre maximal de valeurs différentes de l'observation sur les performances de BaNet3F . . . 156
4.3.3 Influence du nombre de valeurs des champs sur les performances de BaNet3F . . . 157
4.3.4 Influence du nombre maximal de champs par trame sur les performances de BaNet3F . . . 157
4.3.5 Influence du seuil relatif de validation du découpage sur les performances de BaNet3F . . . 161
4.3.6 Influence du nombre maximal d'unités élémentaires considérées simultanément sur les performances de BaNet3F . . . 164
4.3.7 Bilan . . . 167
4.4 Explication de la différence de performances entre traces réelles et générées . . . 169
4.4.1 Hypothèse 1 : différence dans la diversité des valeurs possibles des champs 169

4.4.2 Hypothèse 2 : différence dans la répartition statistique des valeurs possibles des champs . . . 170
4.4.3 Hypothèse 3 : différence dans le nombre de bits fixes . . . 170
4.4.4 Hypothèse 4 : différence dans le niveau de dépendance entre champs . . 171
4.4.5 Hypothèse 5 : différence dans le nombre de champs de 1 bit indépendants 171
4.4.6 Hypothèse 6 : différence dans le nombre de valeurs possibles des champs de 1 bit . . . 172
4.4.7 Bilan sur les hypothèses testées . . . 172
4.5 Conclusion . . . 173

5 Performances comparées des modèles BaNet3F, LDA et de Cai et al. 175
5.1 Performances comparées des modèles BaNet3F et LDA . . . 176
5.1.1 Cadre de la comparaison . . . 176
5.1.2 Performances du modèle LDA . . . 176
5.1.3 Performances du modèle BaNet3F . . . 178
5.1.4 Comparaison des performances des modèles BaNet3F et LDA . . . 180
5.2 Performances comparées des modèles BaNet3F et de Cai et al. . . . 180
5.2.1 Modèle de Cai et al. . . . 181
5.2.2 Cadre des comparaisons . . . 186
5.2.3 Performances du modèle de Cai et al. pour des champs de toutes longueurs 186
5.2.4 Performances du modèle BaNet3F pour des champs de toutes longueurs 188
5.2.5 Comparaison des performances des modèles BaNet3F et de Cai et al. pour des champs de toutes longueurs . . . 189
5.2.6 Performances du modèle de Cai et al. pour des découpages en octets uniquement . . . 190
5.2.7 Performances du modèle BaNet3F pour des découpages en octets uniquement . . . 192
5.2.8 Comparaison des performances des modèles BaNet3F et de Cai et al. pour des découpages en octets uniquement . . . 194
5.3 Conclusion . . . 194

Conclusion générale 197

Annexe 1 : Loi de Dirichlet 201

Annexe 2 : Chaînes de Markov 203

Annexe 3 : Échantillonneur de Gibbs 219

Annexe 4 : Inférence exacte dans les réseaux bayésiens avec l'algorithme *Message Passing* 223

Annexe 5 : Détails sur l'obtention des formules de l'EM appliqué aux réseaux Bayésiens 229

Annexe 6 : Inférence exacte dans les réseaux bayésiens dynamiques avec l'algorithme d'interface 233

Annexe 7 :Détails sur l'obtention des formules de l'EM appliqué aux réseaux Bayésiens dynamiques **241**

Bibliographie **249**

Table des figures

1 Illustration d'un télégraphe de Chappe [1] 8
2 Les principaux systèmes de télécommunication de l'antiquité à nos jours 10
3 Illustration de notre hypothèse de prévisibilité 11
4 Illustration de la pile protocolaire selon la norme OSI dans le cas de l'IoT . . . 11
5 L'importance des WPAN au sein de l'IoT et d'IEEE 802.15.4 au sein des WPAN 14
6 Illustration du scénario de la problématique de la thèse 16
7 Illustration du principe de juxtaposition de piles protocolaires dans l'IoT 19
8 Illustration d'un réseau Bayésien 22

1.1 Procédure de PRE la plus répandue 25
1.2 Trame DLL Enocéan pour une charge utile supérieure à six octets 26
1.3 Structuration hiérarchique d'une population au sens de l'algorithme VDV . . . 27
1.4 Enchaînement des opérations dans l'algorithme VDV 27
1.5 Schéma du principe de fonctionnement de l'algorithme VDV 29
1.6 Exemple de représentation conceptuelle d'une machine d'état de l'algorithme AC 31
1.7 Enchaînement des opérations dans l'algorithme AC 33
1.8 Représentation par plaques du modèle génératif de LDA 37
1.9 Schéma de fonctionnement de l'échantillonneur de Gibbs appliqué au LDA . . . 40

2.1 Principe de génération des trames 46
2.2 Déroulement de l'analyse des trames 48
2.3 Précision de l'algorithme VDV en fonction du nombre de trames et paramétré par la longueur maximale des séquences 52
2.4 Couverture de l'algorithme VDV en fonction du nombre de trames et paramétré par la longueur maximale des séquences 53
2.5 Score F de l'algorithme VDV en fonction du nombre de trames et paramétré par la longueur maximale des séquences 53
2.6 Couverture de l'algorithme VDV en fonction du nombre de trames et paramétré par son nombre de flux 54
2.7 Précision de l'algorithme VDV en fonction du nombre de trames et paramétré par son seuil de filtrage 55
2.8 Couverture de l'algorithme VDV en fonction du nombre de trames et paramétré par son seuil de filtrage 56
2.9 Score F de l'algorithme VDV en fonction du nombre de trames et paramétré par son seuil de filtrage 56
2.10 Précision de l'algorithme AC en fonction du nombre de trames et paramétré par la longueur des séquences 57

2.11 Couverture de l'algorithme AC en fonction du nombre de trames et paramétré par la longueur des séquences 58
2.12 Score F de l'algorithme AC en fonction du nombre de trames et paramétré par la longueur des séquences 59
2.13 Précision de l'algorithme AC en fonction du nombre de trames et paramétré par son seuil de filtrage 59
2.14 Couverture de l'algorithme AC en fonction du nombre de trames et paramétré par son seuil de filtrage 60
2.15 Couverture de l'algorithme AC en fonction du nombre de trames et paramétré par son seuil de filtrage 61
2.16 Précision de l'algorithme AC en fonction du nombre de trames et paramétré par son seuil de fusion 61
2.17 Couverture de l'algorithme AC en fonction du nombre de trames et paramétré par son seuil de fusion 62
2.18 Score F de l'algorithme AC en fonction du nombre de trames et paramétré par son seuil de fusion 63
2.19 Précision de l'algorithme LDA en fonction du nombre de trames et paramétré par la longueur maximale des séquences 63
2.20 Couverture de l'algorithme LDA en fonction du nombre de trames et paramétré par la longueur maximale des séquences 65
2.21 Couverture de l'algorithme LDA en fonction du nombre de trames et paramétré par la longueur maximale des séquences 65
2.22 Précision de l'algorithme LDA en fonction du nombre de trames et paramétré par son nombre de mots-clés 66
2.23 Couverture de l'algorithme LDA en fonction du nombre de trames et paramétré par son nombre de mots-clés 67
2.24 Score F de l'algorithme LDA en fonction du nombre de trames et paramétré par son nombre de mots-clés 67
2.25 Précision de l'algorithme LDA en fonction du nombre de trames et paramétré par son gradient maximal de perplexité 68
2.26 Couverture de l'algorithme LDA en fonction du nombre de trames et paramétré par son gradient maximal de perplexité 68
2.27 Score F de l'algorithme LDA en fonction du nombre de trames et paramétré par son gradient maximal de perplexité 69
2.28 Précision de l'algorithme LDA en fonction du nombre de trames et paramétré par son gradient maximal de probabilité 70
2.29 Couverture de l'algorithme LDA en fonction du nombre de trames et paramétré par son gradient maximal de probabilité 70
2.30 Score F de l'algorithme LDA en fonction du nombre de trames et paramétré par son gradient maximal de probabilité 71
2.31 Précision de l'algorithme LDA en fonction du nombre de trames et paramétré par son paramètre alpha 72
2.32 Couverture de l'algorithme LDA en fonction du nombre de trames et paramétré par son paramètre alpha 72
2.33 Score F de l'algorithme LDA en fonction du nombre de trames et paramétré par son paramètre alpha 73

2.34 Meilleure précision obtenue des algorithmes VDV, AC et LDA en fonction du nombre de trames . . . 75
2.35 Meilleure couverture obtenue des algorithmes VDV, AC, et LDA en fonction du nombre de trames . . . 75
2.36 Meilleur score F obtenu des algorithmes VDV, AC et LDA en fonction du nombre de trames . . . 76
2.37 Meilleur ratio moyen de détection des longueurs obtenu des algorithmes VDV, AC et LDA en fonction du nombre de trames . . . 77
2.38 Meilleur ratio moyen de détection des valeurs obtenu des algorithmes VDV, AC et LDA en fonction du nombre de trames . . . 77
2.39 Meilleur ratio moyen de détection des positions obtenu des algorithmes VDV, AC et LDA en fonction du nombre de trames . . . 78

3.1 Un exemple de réseau Bayésien . . . 83
3.2 Procédure d'inférence exacte dans un réseau Bayésien via *Message Passing* . . . 84
3.3 Trois exemples de graphes . . . 85
3.4 Graphe de la Figure 3.3c moralisé . . . 86
3.5 Exemple de triangularisation . . . 86
3.6 Procédure d'élimination des variables appliquée au graphe de la Figure 3.5b . . . 88
3.7 *junction tree* du graphe de la Figure 3.3c . . . 88
3.8 Transmission des messages dans le *Message Passing* pour le *junction tree* de la Figure 3.7 . . . 89
3.9 Exemple de détermination de la frontière de Markov . . . 93
3.10 Exemple de "découpage" à l'aide de la frontière de Markov . . . 94
3.11 Découpage d'un réseau en suite de frontières de Markov . . . 95
3.12 Représentation classique, dite "2 tranches" d'un RBD générique . . . 99
3.13 Procédure d'inférence exacte dans un réseau Bayésien dynamique avec l'algorithme d'interface . . . 100
3.14 Mise en évidence des interfaces *Forward* I_t^{t+1} et *Backward* I_t^{t-1} sur un exemple 101
3.15 Représentation "1 tranche" du graphe de la Figure 3.14 moralisé et triangularisé 101
3.16 *junction tree* de la tranche temporelle t du réseau exemple . . . 101
3.17 Principe de connexion des *junction tree*s pour un réseau d'ordre N . . . 102
3.18 Principe de connexion des *junction tree*s pour un réseau d'ordre un . . . 103
3.19 Un *junction tree* J_t non terminal de notre exemple et ses connexions à ses voisins 103
3.20 Exemple de couverture de Markov évidente d'un réseau Bayésien dynamique . . . 106
3.21 Mise en évidence des tranches de définition sur un exemple . . . 107
3.22 Vue globale de BaNet3F . . . 110
3.23 Vue globale de l'opération d'analyse effectuée par le modèle . . . 111
3.24 Réseau Bayésien utilisé dans le modèle . . . 112
3.25 Réseau de référence . . . 115
3.26 Algorigramme du déroulement général de l'algorithme AMBV . . . 123
3.27 Frontières de Markov du réseau de BaNet3F générées avec l'algorithme . . . 123

4.1 Un exemple de réseau utilisé pour générer des trames de 6 champs . . . 133
4.2 Un exemple de distribution de champ initial . . . 133
4.3 Un exemple de distribution de champ non-initial . . . 134

4.4 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur tous les découpages . . . 137
4.5 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur tous les découpages, et lissé sur 5 points . . . 138
4.6 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les champs de moins d'un octet . . . 139
4.7 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les champs de moins d'un octet, et lissé sur 5 points . . . 139
4.8 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les champs d'un octet ou plus . . . 140
4.9 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages mélangeant des champs de toutes longueurs . . . 140
4.10 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages simples . . . 141
4.11 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages complexes . . . 142
4.12 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages complexes, et lissé sur 5 points . . . 142
4.13 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages trouvables pour 2 champs maximal par trame ou plus . . . 143
4.14 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages trouvables pour 2 champs maximal par trame ou plus, et lissé sur 5 points . . . 143
4.15 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages trouvables pour 3 et 4 champs maximal par trame ou plus . . . 144
4.16 Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur tous les découpages, et lissé sur 5 points . . . 145
4.17 Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les champs de moins d'un octet, et lissé sur 5 points . . . 145
4.18 Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les champs d'un octet ou plus . . . 146
4.19 Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages mélangeant des champs de toutes longueurs, et lissé sur 3 points . . . 146
4.20 Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages simples . . . 147
4.21 Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages complexes, et lissé sur 5 points . . . 148
4.22 Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages trouvables pour 2 champs maximal par trame ou plus, et lissé sur 5 points . . . 148
4.23 Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages trouvables pour 3 et 4 champs maximal par trame ou plus . . . 149

4.24 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur tous les découpages, et lissé sur 5 points 149
4.25 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur les champs de moins d'un octet, et lissé sur 3 points 150
4.26 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur les champs d'un octet ou plus 150
4.27 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur les découpages simples, et lissé sur 3 points 151
4.28 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur les découpages complexes, et lissé sur 5 points 152
4.29 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur les découpages trouvables pour 2 champs maximal par trame ou plus, et lissé sur 5 points 152
4.30 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur les découpages trouvables pour 3 et 4 champs maximal par trame ou plus, et lissé sur 5 points 153
4.31 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMVDO, moyenné sur tous les découpages, et lissé sur 3 points 156
4.32 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMVDO, moyenné sur les découpages longs 157
4.33 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NVC, moyenné sur tous les découpages, et lissé sur 3 points 158
4.34 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NVC, moyenné sur les découpages longs 158
4.35 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NVC, moyenné sur les découpages courts 159
4.36 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur tous les découpages, et lissé sur 3 points 159
4.37 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages longs 160
4.38 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages courts 160
4.39 Précision du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur tous les découpages, et lissé sur 3 points 161
4.40 Couverture du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur tous les découpages, et lissé sur 3 points 162
4.41 Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur tous les découpages, et lissé sur 3 points 162
4.42 Précision du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages longs 163
4.43 Couverture du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages longs 163
4.44 Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages longs 164
4.45 Précision du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages courts 165

4.46 Couverture du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages courts . . . 165
4.47 Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages courts . . . 166
4.48 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur tous les découpages, et lissé sur 3 points . . . 166
4.49 Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur les découpages longs . . . 167
4.50 Comparaison du Score F du modèle BaNet3F en fonction du RDM entre traces réelles et générées, moyenné sur tous les découpages . . . 168

5.1 Score F du modèle LDA en fonction du ratio de dépendances moyennes, paramétré par le nombre de mots-clefs, et lissé sur 5 points . . . 177
5.2 Score F du modèle LDA en fonction du ratio de dépendances moyennes, paramétré par le gradient maximal de probabilité, et lissé sur 5 points . . . 178
5.3 Score F du modèle BaNet3F en fonction du ratio de dépendances moyennes, paramétré par le nombre maximal de champs par trame, et lissé sur 5 points . . . 179
5.4 Score F du modèle BaNet3F en fonction du ratio de dépendances moyennes, paramétré par le seuil relatif de validation du découpage, et lissé sur 5 points . . . 180
5.5 Meilleurs scores F des modèles LDA et BaNet3F en fonction du ratio de dépendances moyennes . . . 181
5.6 Vue d'ensemble de la procédure d'apprentissage des formats de trames . . . 182
5.7 Illustration de la durée variable d'un état de la variable cachée . . . 183
5.8 Représentation sous forme graphique du modèle de génération des trames . . . 184
5.9 Exemple d'état du modèle lors de la génération d'une trame . . . 185
5.10 Exemple de trame HTTP segmentée en champs à l'aide de l'algorithme de Viterbi 185
5.11 Score F du modèle Cai et al. lorsqu'appliqué à tous les découpages, en fonction du ratio de dépendances moyennes, paramétré par le seuil de filtrage des séquences fermées récurrentes, et lissé sur 5 points . . . 187
5.12 Score F du modèle BaNet3F lorsqu'appliqué à tous les découpages, en fonction du ratio de dépendances moyennes, paramétré par le nombre maximal de champs par trame, et lissé sur 5 points . . . 189
5.13 Score F du modèle BaNet3F lorsqu'appliqué à tous les découpages, en fonction du ratio de dépendances moyennes, paramétré par le seuil relatif de validation du découpage, et lissé sur 5 points . . . 190
5.14 Meilleurs scores F des modèles Cai et al. et BaNet3F lorsqu'appliqués à tous les découpages, en fonction du ratio de dépendances moyennes . . . 191
5.15 Score F du modèle Cai et al. lorsqu'appliqué aux découpages en octets, en fonction du ratio de dépendances moyennes, paramétré par le seuil de filtrage des séquences fermées récurrentes . . . 192
5.16 Score F du modèle BaNet3F lorsqu'appliqué aux découpages en octets, en fonction du ratio de dépendances moyennes, paramétré par le nombre maximal de champs par trame . . . 193
5.17 Score F du modèle BaNet3F lorsqu'appliqué aux découpages en octets, en fonction du ratio de dépendances moyennes, paramétré par le seuil relatif de validation du découpage . . . 194

5.18 Meilleurs scores F des modèles Cai et al. et BaNet3F lorsqu'appliqués aux découpages en octets, en fonction du ratio de dépendances moyennes 195
5.19 Mise en perspective du travail effectué dans le cadre de la thèse par rapport à la démarche globale proposée 199
5.20 Exemple de Modèle de Markov à deux états 204
5.21 Exemple de Modèle de Markov Caché à deux états et deux observations 205
5.22 Illustration graphique de l'algorithme *Forward-Backward* 215
5.23 Illustration graphique de l'algorithme de Viterbi 217
5.24 Exemple d'approximation de rapport de surfaces via simulation de Monte-Carlo 219
5.25 Schéma de fonctionnement de l'échantillonneur de Gibbs 221
5.26 Schéma de fonctionnement de l'échantillonneur de Gibbs générique 221
5.27 Schéma du réseau bayésien utilisé pour l'exemple de *Message Passing* 223
5.28 Affectation des potentiels initiaux des cliques et des séparateurs 224
5.29 Schéma du réseau bayésien dynamique utilisé pour l'exemple d'algorithme interface 233
5.30 Affectation des potentiels initiaux des cliques et des séparateurs 235

Remerciements

Je tiens à remercier mon directeur de thèse, M. Louet, qui a su me guider tout au long de ces quatre années, et aussi me remonter le moral lorsque les résultats n'étaient pas à la hauteur de mes espérances.

Je suis également très reconnaissant à mon co-encadrant de thèse, M. Djoko-Kouam, qui était toujours là pour répondre au jour le jour à mes questions et m'aider à surmonter les divers points durs rencontrés.

Je remercie les membres de mon jury de thèse pour avoir accepté de prendre le temps de lire et donner leur avis sur le présent manuscrit, ainsi que de se rendre sur le lieu de la soutenance.

Je tiens aussi à remercier mes parents pour l'important soutien logistique qu'il m'ont apporté durant ces quatre années. Grâce à eux, j'ai pu pleinement me consacrer à ma thèse sans me soucier du reste.

M. Fougères, le directeur de mon laboratoire de recherche, a su être de bon conseil à mon égard, et m'ouvrir à lui de mes avancées et problèmes au hasard de nos rencontres m'a parfois aidé à prendre du recul pour mieux repartir. Pour cela, je le remercie.

Merci à M. Boiteau, avec qui j'ai pu régulièrement discuter de tout et de rien pour me changer les idées lorsque j'étais bloqué, et qui au travers du partage de son expérience a parfois pu me débloquer.

Je remercie M. Guérin, qui m'a apporté un important soutien logistique sur le plan du hardware et du software informatique, sans lequel j'aurai eu bien du mal à mener à bien toutes les simulations qui ont été nécessaires à cette thèse.

Je remercie également M. Bonnin et M.Billot, les deux membres de mon CSI, qui m'ont vraiment aidé à cadrer ma thèse tout au long de nos trois réunions. Grâce à eux j'ai pu me redéfinir un objectif clair à chaque fois que je commençais à me disperser.

Je souhaite adresser un grand merci à Mme Courbin, responsable de l'enseignement de l'anglais à l'ECAM, pour m'avoir aidé à corriger mes diverses publications internationales.

Merci aussi à M. Elvira, enseignant à CentraleSupélec et membre de mon équipe de recherche, SCEE, d'avoir bien voulu me consacrer un peu de son temps pour me donner son avis sur mes recherches. Bien que nous ayons quelque peu manqué de temps, il a su me faire quelques suggestions qui ont abouti à des idées m'ayant été utile par la suite.

Je tiens également à adresser mes remerciements à M. Maître, directeur de l'ECAM, pour avoir accepté de subventionner ma thèse via les fonds de l'ECAM, et de m'avoir accueilli au sein de son établissement durant ces quatre années.

Enfin, je remercie toutes les personnes ayant contribué directement ou indirectement à l'aboutissement de cette thèse.

Abréviations

AC	Aho-Corasick
$AMBV$	Automatic Markov Boundaries construction optimized Viterbi
$BaNet3F$	Bayesian Network Frame Format Finder
CM	Chaînes de Markov
DLL	Data Link Layer (Couche Liaison de Données)
ESS	Estimation de la Statistique Suffisante
GDA	Graphe Direct Acyclique
ISO	International Standardization Organization
LDA	Latent Dirichlet Allocation
LTE	Long-Term Evolution
$NMCT$	Nombre maximal de champs par trame
$NMUECS$	Nombre maximal d'unités élémentaires considérées simultanément
$NMVDO$	Nombre maximal de valeurs différentes de l'observation
NVC	Nombre de valeurs des champs
OSI	Open Systems Interconnection
PRE	Protocol Reverse Engineering
RB	Réseaux Bayésiens
RBD	Réseaux Bayésiens Dynamiques
RDM	Ratio de Dépendances Moyennes
RIP	Running Intersection Property
RL	Radio Logicielle
$SRVD$	Seuil relatif de validation du découpage
SVM	Support Vector Machine
VDV	Variance Distribution of the Variances
$WPAN$	Wireless Personal Area Network

Liste des publications de l'auteur dans le cadre de la thèse

Conférences internationales

— Pierre-Samuel Gréau-Hamard, Moïse Djoko-Kouam, Yves Louet. A comparative Study of Sequence Identification Algorithms in IoT Context. 2^{nd} *International Conference on Advances in Signal Processing and Artificial Intelligence (ASPAI' 2020)*, November 2020 *(accepté)*.

— Pierre-Samuel Gréau-Hamard, Moïse Djoko-Kouam, Yves Louet. AMBV : An Optimized Generic Viterbi Algorithm for Bayesian Networks. *International Conference on Big data, Machine Learning and IoT Applications (ICBMLIA-2021)*, December 2021 *(accepté)*.

— Pierre-Samuel Gréau-Hamard, Moïse Djoko-Kouam, Yves Louet. A Bayesian Networks based Automatic Binary Frame Format Identification Model : BaNet3F. *3rd URSI Atlantic / Asia-Pacific Radio Science Meeting (AT-AP-RASC 2022)*, May 2022 *(accepté)*.

Journaux scientifiques

— Pierre-Samuel Gréau-Hamard, Moïse Djoko-Kouam, Yves Louet. Performance Analysis and Comparison of Sequence Identification Algorithms in IoT Context. *Sensors & Transducers*, vol. 249, no. 2, February 2021 *(accepté)*.

— Pierre-Samuel Gréau-Hamard, Moïse Djoko-Kouam, Yves Louet. AMBV : An Optimized Generic Viterbi Algorithm for Bayesian Networks. *Journal of Scientometric Research (J. Scientometric Res.)(accepté)*

— Pierre-Samuel Gréau-Hamard, Moïse Djoko-Kouam, Yves Louet. BaNet3F : A Bayesian Network Approach to Automatic Binary Frame Format Identification. *International Journal of Big Data, AI & IoT (IJBDAI)(accepté)*.

Introduction

1 Les origines des télécommunications et des objets connectés

1.1 Des origines du besoin de télécommuniquer

L'une des principales caractéristiques de l'espèce humaine, à la différence de toute autre sur Terre, réside dans sa capacité à communiquer de façon poussée. Bien que la plupart des animaux communiquent entre eux, la quantité d'information échangée n'a aucune commune mesure avec la nôtre. Grâce à cette particularité, toute connaissance acquise par un individu est transmissible au reste de la population, et toute tâche à effectuer par le groupe est répartissable entre tous ses individus, permettant d'améliorer l'efficacité du groupe. Il s'agit de l'une des principales raisons ayant permis à l'humanité de se développer aussi vite.

Les humains se sont assez rapidement rendu compte que, seul, un individu est relativement faible, mais en tant que groupe, grâce à la communication entre ses membres, peu de choses lui sont impossibles. Le langage a donc été perfectionné, ce qui a permis aux groupes d'humains de grandir et d'étendre leurs territoires.

Cependant, à ce stade, les limites de la voix humaine se sont fait sentir : bien que sa vitesse de propagation soit relativement élevée, sa portée reste très faible. Une nouvelle problématique est alors apparue : comment maintenir la communication entre les individus malgré la distance ? La solution la plus évidente, et celle utilisée initialement, est d'utiliser des messagers, transportant physiquement l'information. Toutefois, même si cela permet de communiquer avec plusieurs entités à distance simultanément, le temps de propagation est très élevé. C'est en cherchant à apporter une réponse plus efficace à cette problématique qu'est né le domaine des télécommunications.

1.2 Les premiers systèmes de télécommunication

Les premiers systèmes de communication longue distance avaient tous un point commun : ils utilisaient la propagation de la lumière comme vecteur de transmission. La première mention de l'emploi d'un dispositif de télécommunication à longue distance remonte à la Grèce Antique, durant la Guerre de Troie, au XII^e siècle avant J.C., où l'annonce de la victoire est envoyée jusqu'à Argos dans le Péloponnèse, à 700km de distance. Cette communication est réalisée via des tours situées de lieu en lieu sur des points hauts, sur lesquelles étaient allumés un certain nombre de feux correspondant à un message particulier [2, 3].

L'autre type de système aussi utilisé depuis une époque reculée consistait à utiliser non

pas directement les feux, mais la fumée qu'ils génèrent lorsque l'on y fait brûler certaines substances, comme de l'herbe ou du bois vert. Ce système était notamment employé par les Amérindiens bien avant l'arrivée des grands explorateurs.

Ces systèmes, bien qu'accélérant fortement la propagation de l'information comportaient une limite majeure : la complexité du signal était trop faible pour pouvoir porter un message élaboré. Pour faire un parallèle avec la quantification actuelle de l'information numérique, les messages ne comportaient que quelques bits, moins d'un octet.

1.3 Les réseaux de télécommunication modernes

C'est pour repousser cette limite qu'un nouveau type de transmission de données vit le jour à la fin du $XVIII^e$ siècle : le télégraphe de Chappe. Ce dispositif consistait en un enchaînement de tours relais disposées en hauteur de lieu en lieu, comme pour les dispositifs précédents. Chacune de ces tours comportait un mat sur lequel étaient accrochées trois pièces articulées pouvant prendre différentes positions, chacune de celles-ci correspondant à une information spécifique, comme illustré sur la Figure 1. L'avantage de ce système par rapport aux précédents est la possibilité d'utiliser un enchaînement de ces différentes positions afin de créer un message complexe, pouvant typiquement contenir jusqu'à l'équivalent de plusieurs centaines d'octets d'information [2]. Ce système présentait cependant encore un certain nombre de limitations. La plus importante est qu'il ne fonctionnait que le jour et par temps clair. De plus, le temps de transmission était assez important, à cause de la reformulation complète du message nécessaire à chaque relais. Enfin, la présence continue d'un opérateur dans chaque relai était indispensable, et le risque de défaillance humaine était donc constamment présent.

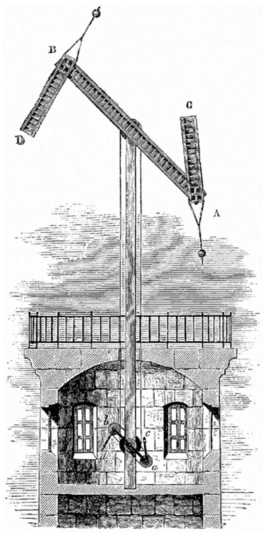


FIGURE 1 – Illustration d'un télégraphe de Chappe [1]

Au début du XIX^e siècle, l'électricité fit son apparition [4], et avec elle s'ouvrit de nouvelles perspectives, notamment dans le domaine des télécommunications, en offrant un nouveau vecteur de transmission à l'information. C'est ainsi que le télégraphe de Morse vit le jour au milieu de ce siècle résolvant tous les problèmes précédemment cités du télégraphe de Chappe [2]. La taille et la complexité des messages restaient les mêmes que pour ce dernier, mais la transmission devenait quasi instantanée. Quelques décennies plus tard, en 1880, arrivait le téléphone, extension du télégraphe à la voix, avec un débit et une complexité de l'information jamais atteint auparavant. Il restait encore toutefois quelques limitations : le besoin d'opérateurs pour mettre en communication les utilisateurs en reliant leurs lignes respectives, et la nécessité d'installer d'importantes infrastructures sous la forme de milliers de kilomètres de lignes.

Au début du XX^e siècle, la première transmission radio longue distance fut réalisée par Guglielmo Marconi entre le Canada et l'Angleterre. Le besoin d'infrastructure se retrouvait donc fortement réduit, se limitant à des antennes ponctuelles, permettant de déployer un réseau de grande envergure bien plus rapidement et à moindre coût. Ainsi la radio se répandit à grande

échelle à parti des années 1920, rejointe dans les années 1930 par la télévision. Désormais, le débit d'information était suffisant pour transmettre à la fois du son et de l'image en temps réel [2].

1.4 Le développement de la téléphonie mobile

Désormais, tous les éléments étaient réunis pour les débuts de la téléphonie mobile, ou 0G, qui apparut à la fin des années 1940. Le dispositif nécessaire était effectivement transportable, mais son encombrement et son besoin en énergie le rendait dépendant d'un élément externe : l'automobile. De plus, une fréquence donnée ne pouvait supporter qu'une seule communication à la fois, et chaque cellule couvrant le territoire n'était déservie que par une unique station de base de forte puissance, n'autorisant pas les déplacements de l'une à l'autre au cours d'une communication [2, 5]. Ce n'est que dans les années 1970 qu'arrivent les premiers téléphones réellement portables, entièrement autonomes et tenant dans la main (bien que restant assez lourds et encombrants).

A la fin des années 1970 arrive la 1G, offrant des réseaux cellulaires permettant un déplacement d'une cellule à l'autre, ou mécanisme de handover. Avec l'apparition de la 2G au début des années 1990, les portables s'allègent et adoptent un facteur de forme proche de celui actuel, les débits augmentent, s'élevant à environ 10kb/s, et il devient possible de transmettre des messages textuels en plus de la voix [6]. Vient ensuite la 3G au début des années 2000, qui améliore encore les débits de transmission, les multipliant environ par 100, et offre désormais l'accès à Internet. Avec l'arrivée de la 4G au début des années 2010, les débits sont une nouvelle fois multipliés par 100, atteignant les 100Mb/s.

Au cours de toutes ces évolutions, on peut voir que l'accent a été principalement mis sur l'augmentation du débit disponible pour chaque terminal plus que sur le nombre de terminaux pouvant être géré par le réseau. En effet, entre l'arrivée de la 2G et celle de la 4G, les débits ont été multipliés par environ 10 000, tandis que le nombre d'utilisateurs a été multiplié par 200 dans le même temps en France [7].

1.5 L'arrivée des objets connectés et la 5G

Cependant, entre l'arrivée de la 3G et la 4G, un nouveau concept a fait son apparition au milieu des années 2000 : les objets connectés [8]. On entend par là des objets ayant la capacité de communiquer en réseau, généralement via liaison radio, reliés à Internet ou non, et voués à être présents en grands nombres. Bien qu'assez peu adoptés les premières années, les objets connectés voient leur nombre entamer une croissance fulgurante à la fin des années 2000, bien plus rapide que celle de tous les autres types d'appareils reliés à Internet, comme les ordinateurs ou les smartphones. On attend à l'heure actuelle environ 75 milliards d'objets connectés d'ici 2025 [9].

Les objets connectés ont des besoins en connectivité beaucoup plus variés que ceux des appareils connectés classiques, parfois même totalement opposés, à savoir aucun besoin d'un débit important, mais la nécessité d'une prise en charge par le réseau d'un très grand nombre d'entre eux, tandis que les ordinateurs et smartphones ont besoin d'un fort débit, mais sont nettement moins nombreux à souhaiter se connecter au réseau. Le modèle d'évolution des

télécommunications sans fil jusqu'à présent, à savoir, forte augmentation du débit disponible pour chaque utilisateur, et légère augmentation du nombre maximal d'utilisateurs n'est donc plus viable, car ce serait laisser de côté les objets connectés. C'est pourquoi la nouvelle norme en cours de déploiement depuis la fin des années 2010, la 5G, offrira, entre autres, à la fois des débits multipliés par 100 par rapport à la 4G de base, et un nombre de terminaux connectés simultanément multiplié par 100 [10].

1.6 Bilan

Les dates et systèmes de télécommunication importants sont rassemblés sur la frise Figure 2. On peut constater une accélération évidente du rythme de développement, passant de plusieurs siècles entre chaque système à quelques décennies, puis quelques années. La compacité des systèmes elle aussi a fortement évoluée ; de bâtiments entiers au départ, ils ont été miniaturisés jusqu'à tenir dans une main. Enfin, les débits ont crus de quelques octets par heure jusqu'à plusieurs milliards d'octets par seconde.

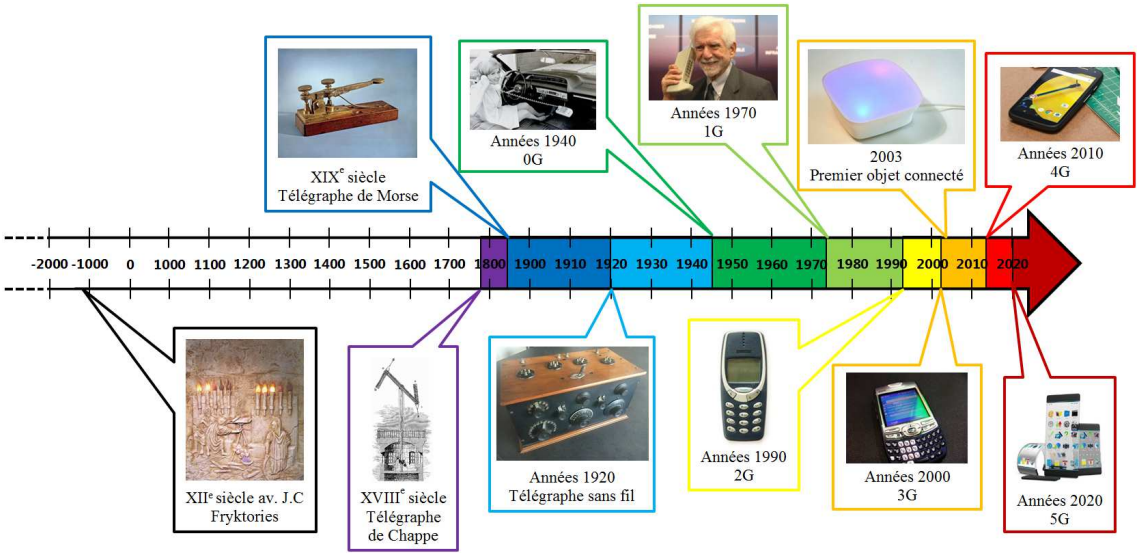


FIGURE 2 – Les principaux systèmes de télécommunication de l'antiquité à nos jours

Parmi tous les systèmes communicants existant, dans cette thèse, on se concentrera sur les réseaux d'objets connectés, et plus particulièrement leurs protocoles de communication.

2 De l'imprévisibilité du protocole d'une couche sachant ses couches inférieures

Pour répondre aux besoins de communication grandissant des objets communicants, et aux applications très diverses demandant des performances de plus en plus importantes, nombre de protocoles voient le jour chaque année [11]. Ces protocoles sont pour l'essentiel basés sur le modèle OSI (Open Systems Interconnection) développé par l'organisation ISO (International Standardization Organization) [12] afin de permettre l'interopérabilité des différentes parties d'une pile protocolaire. Avec ce principe d'interopérabilité des couches d'une pile protocolaire, la connaissance du protocole d'une couche donnée n'induit donc pas toujours la connaissance

du reste de la pile. Nous allons donc prouver que, dans le cas de l'IoT, sachant les protocoles des couches 1 à n-1 d'une pile protocolaire, il n'est pas toujours possible de prévoir [1] le protocole de la couche n de cette même pile, comme illustré sur la Figure 3.

2.1 Constitution d'un échantillon de protocoles

Nous avons constitué un échantillon de 24 protocoles multi-couches couramment utilisés dans l'IoT afin d'avoir une vue d'ensemble de la constitution des piles protocolaires dans l'IoT. Afin de nous assurer que cet échantillon soit représentatif, nous avons sélectionné aléatoirement des références présentant les protocoles multi-couches les plus utilisés dans l'IoT. Pour chacune de ces références, nous avons dénombré les protocoles présents dans notre échantillon et ceux ne l'étant pas. Nous obtenons une proportion moyenne de protocoles présents par rapport à tous ceux cités de 80% environ. Nous avons ensuite recensé tous les protocoles cités au moins une fois dans les références et faisant partie de l'échantillon, ainsi que ceux n'en faisant pas partie. Même si seulement 70% des protocoles de notre échantillon sont cités, ils représentent tout de même 60% des protocoles cités. Nous pouvons donc conclure que notre échantillon comprend au moins la moitié des protocoles connus de l'IoT, et donc qu'il est suffisamment représentatif pour servir de base au raisonnement qui suivra.

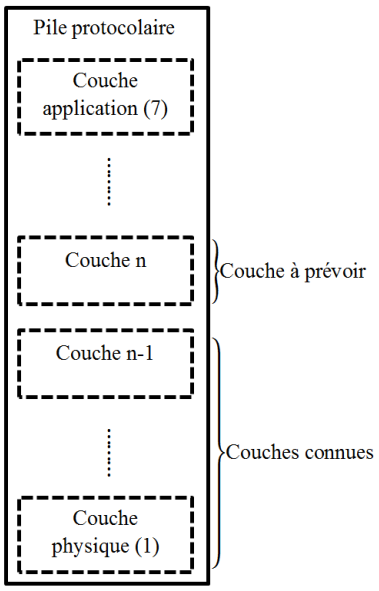


FIGURE 3 – Illustration de notre hypothèse de prévisibilité

La pile protocolaire d'un objet communicant est fortement inspirée du modèle OSI, mais simplifiée avec la fusion des couches transport et réseau, et la suppression des couches session et présentation. La pile se résume donc aux couches Physique, Liaison de données, Réseau, et Application, comme illustré sur la Figure 4.

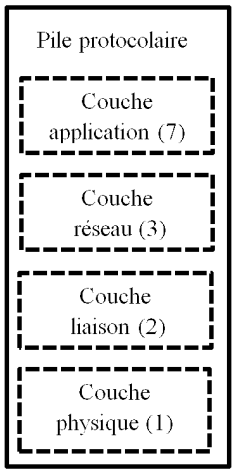


FIGURE 4 – Illustration de la pile protocolaire selon la norme OSI dans le cas de l'IoT

L'échantillon de protocoles est présenté dans la Table 1. Certains protocoles, notamment ceux issus de la famille Long-Term Evolution (LTE) ne sont pas représentés dans cet échantillon par manque d'informations concises.

L'indication 'Libre' signifie que la documentation du protocole considéré n'inclut pas cette couche. L'indication 'Propriétaire' signifie que la couche est définie par le protocole considéré, mais qu'elle n'a pas été normalisée. L'indication 'quelconque' signifie que la norme de la couche

1. La prévisibilité de protocole de couche dont nous parlons ne concerne que la partie transmission de données d'une pile à une autre, en excluant les communications inter-couches. Cela vaut aussi bien pour la couche à prévoir que pour les couches considérées connues.

	Protocoles	Couche application	Couche réseau	Couche liaison de données	Couche physique
1	6LoWPAN [13]	Libre	IPv6 with LoWPAN	IEEE 802.15.4 (quelconque)	IEEE 802.15.4-2006 (quelconque)
2	ANT [14]	Libre	Propriétaire	Propriétaire	Propriétaire
3	Bluetooth smart [15]	Libre	Libre	IEEE 802.15.1	IEEE 802.15.1
4	Dash7 [16]	ISO/IEC 18000-7	ISO/IEC 18000-7	ISO/IEC 18000-7	ISO/IEC 18000-7
5	EnOcean [17]	Propriétaire	ISO/IEC 14543-3-1X	ISO/IEC 14543-3-1X	ISO/IEC 14543-3-1X
6	Insteon [15]	Propriétaire	Propriétaire	Propriétaire	Propriétaire
7	ISA100.11a [18]	Propriétaire	IPv6 with LoWPAN	IEEE 802.15.4-2006 (customisé)	IEEE 802.15.4-2006 2,4GHz O-QPSK
8	KNX [19]	Libre	Propriétaire	IEC 870-5-2	ISO/IEC 14443-2
9	LoRaWAN [15]	Libre	Libre	Propriétaire	Propriétaire
10	MiWi [20]	Libre	Propriétaire	IEEE 802.15.4-2006 (customisé)	IEEE 802.15.4-2006 2,4GHz O-QPSK
11	NB-Fi [21]	Propriétaire	Propriétaire	Propriétaire	Propriétaire
12	Nwave [15]	Propriétaire	Propriétaire	Propriétaire	Propriétaire
13	Rubee [22]	IEEE 1902.1	IEEE 1902.1	IEEE 1902.1	IEEE 1902.1
14	Sigfox [17]	Libre	Propriétaire	Propriétaire	Propriétaire
15	Thread [23]	Libre	IPv6 with LoWPAN	IEEE 802.15.4-2006 (customisé)	IEEE 802.15.4-2006 2,4GHz O-QPSK
16	Weightless P, W [15]	Libre	Propriétaire	Propriétaire	Propriétaire
17	WiFi WAVE [24]	Libre	IEEE 1609.3	IEEE 802.11 p, IEEE 1609.4, IEEE 802.2	IEEE 802.11 p
18	WiFi-ah (HaLow)	Libre	Libre	IEEE 802.11 ah	IEEE 802.11 ah
19	Wimax [25]	Libre	Libre	IEEE 802.16	IEEE 802.16
20	Wireless M-BUS [26]	EN-13757	EN-13757	EN-13757	EN-13757
21	WirelessHART [18]	Propriétaire	Propriétaire	IEEE 802.15.4-2006 (customisé)	IEEE 802.15.4-2006 2,4GHz O-QPSK
22	Wi-SUN [27]	Divers (4 variantes, dépendant de la couche réseau)	Divers (4 variantes)	IEEE 802.15.4/4e-2006 (quelconque)	IEEE 802.15.4-2006g (quelconque)
23	Zigbee [19]	Propriétaire	Propriétaire	IEEE 802.15.4-2003 (customisé)	IEEE 802.15.4-2003 (quelconque)
24	Z-Wave [17]	Propriétaire	Propriétaire	ITU-T G.9959	ITU-T G.9959

TABLE 1 – Piles protocolaires de l'échantillon de protocoles de l'IoT

offre plusieurs alternatives, et qu'elles peuvent être indifféremment utilisées dans le protocole considéré. L'indication 'customisé' signifie que la couche s'est fortement inspirée de la norme, à quelques variantes près.

Pour chaque couche de la pile protocolaire, on estime statistiquement la probabilité de pouvoir prévoir le protocole de la couche considérée sachant ceux de ses couches inférieures. Pour ce faire, on calcule le ratio entre le nombre de protocoles pour lesquels il est possible de prévoir la couche considérée simplement en connaissant ses couches inférieures, et le nombre total de protocoles. Les résultats obtenus sont présentés dans la Table 2.

	Couche application	**Couche réseau**	**Couche liaison de données**	**Couche physique**
Probabilité de ne pas pouvoir prévoir le protocole de la couche	50%	21%	25%	100%
Probabilité de pouvoir prévoir le protocole de la couche	50%	79%	75%	0%

Table 2 – Probabilité de prédiction du protocole d'une couche

Par exemple, si l'on prend la couche physique du protocole Thread, on ne peut pas prévoir la couche liaison de données car cette couche physique particulière (IEEE 802.15.4-2006 2,4GHz O-QPSK) est aussi employée par 6LoWPAN, ISA100.11, MiWi, WirelessHART, et Wi-SUN, protocoles qui ont des couches liaison de données différentes. À contrario, si l'on prend la couche physique du protocole DASH7, on peut prévoir la couche liaison de données car pour cette couche physique particulière (ISO/IEC 18000-7), une seule couche liaison de données est présente dans notre échantillon.

2.2 Prévisibilité du protocole de la couche liaison de données sachant celui de la couche physique

À partir de notre échantillon ainsi constitué, ainsi que de sources externes, nous allons montrer qu'il n'est pas toujours possible de prévoir le protocole de la couche liaison de données sachant celui de la couche physique. Nous allons pour cela d'abord interpréter les résultats obtenus sur notre échantillon de protocoles, puis présenter la notion de WPAN, dont nous aurons besoin pour la suite. Subséquemment, nous exposerons la place de la norme IEEE 802.15.4 au sein des protocoles de l'IoT, et enfin les perspectives de développement de cette norme.

Interprétation des résultats sur notre échantillon de protocoles

Nous constatons à l'aide de la Table 1 que, connaissant la couche physique, la couche liaison de données n'est pas prévisible dans 6 cas sur 24 dans notre échantillon, soit 25% des cas. Nous remarquons qu'il s'agit de ceux pour lesquels ces deux couches sont issues de la norme IEEE 802.15.4. Cette imprévisibilité s'explique par la multiplicité des couches PHY et des

mécanismes MAC proposés par la norme, permettant le choix de nombreuses combinaisons par les concepteurs de chaque protocole. Ces cas, qui nous intéressent, sont ceux pour lesquels la couche supérieure n'est pas prévisible.

Le fait que l'espace d'imprévisibilité de la couche liaison de données se limite à la norme IEEE 802.15.4 peut sembler relativement problématique, cependant nous allons montrer que cet ensemble est de taille conséquente, et amené à grossir dans le futur. Notre travail se concentrera donc principalement sur les Wireless Personal Area Networks (WPANs), et plus particulièrement sur la niche que constitue la famille des protocoles basés sur IEEE 802.15.4, dans laquelle on retrouve le plus fort taux d'imprévisibilité.

Les WPANs

Les WPANs sont un type de réseaux offrant des gammes de débit allant de l'ordre du ko/s au Mo/s, et permettant une communication sur une faible portée (de 10 à 100m) pour une très faible consommation (quelques mW) [28].

Ils sont principalement dédiés à la maison connectée, mais peuvent aussi être utilisés dans différents domaines, notamment l'industrie, avec par exemple WirelessHART et ISA100.11a ayant été développés dans ce but.

Le concept de WPANs a été conçu par le groupe de travail 802.15 de l'IEEE. Au sein de ce groupe existent 12 sous-groupes, dont les plus connus sont 802.15.1, ayant normalisé le Bluetooth, et 802.15.4, ayant développé la norme fondatrice des Low Rate-WPANs.

La norme IEEE 802.15.4 est certainement la plus dynamique et évolutive au sein des protocoles des WPAN, avec 4 versions à ce jour, 2 amendements à la dernière version publiés, et 6 projets d'amendements supplémentaires en cours [29, 30].

La place d'IEEE 802.15.4 au sein des protocoles de l'IoT

À l'heure actuelle, les protocoles des WPANs représentent la majeure partie de ceux implémentés sur les objets connectés [31]. Si l'on regarde notre échantillon, on constate que 14 sur les 24 protocoles sont utilisés dans des WPANs, soit environ 60%, ce qui est cohérent avec cette tendance. Ces chiffres sont illustrés sur la Figure 5.

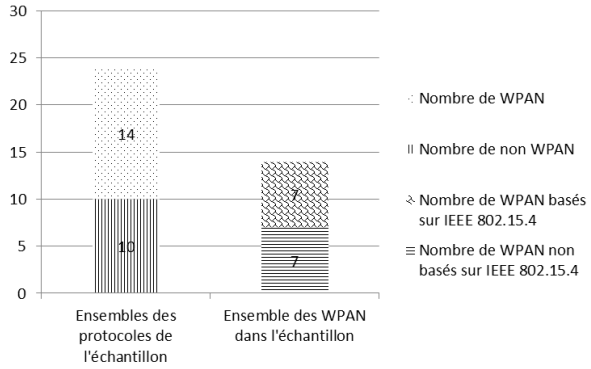


FIGURE 5 – L'importance des WPAN au sein de l'IoT et d'IEEE 802.15.4 au sein des WPAN

Actuellement, les protocoles les plus utilisés dans les WPANs sont BLE, Zigbee, Wi-Fi, Z-Wave, et Thread [32]. De ces 5 protocoles, 2 sont basés sur IEEE 802.15.4, soit 40%, ce qui correspond environ à la proportion générale de la norme au sein des protocoles de notre échantillon. Si l'on reprend ce dernier, on se rend compte que 7 protocoles sur les 14 utilisés dans les WPAN, soit 50%, sont basés sur

IEEE 802.15.4, ce qui confirme que cette norme est la plus répandue dans les WPANs [33], comme illustré sur la Figure 5.

Nous avons vu que la norme IEEE 802.15.4 constitue une part importante des protocoles des WPANs, et que ceux-ci sont majoritaires au sein de l'IoT. Nous pouvons donc en conclure que la norme IEEE 802.15.4 est d'importance majeure au sein de l'IoT.

Les perspectives de développement d'IEEE 802.15.4

Deux des protocoles leaders dans les WPAN, l'un des plus anciens, Zigbee (2007), et le plus récent, Thread (2015) s'apprêtent à fusionner [34]. En effet, la Zigbee Alliance et le Thread Group prévoient d'utiliser les couches basses du protocole Thread avec la couche application du protocole Zigbee afin de créer un nouveau protocole. L'alliance de ces deux poids lourds du marché basés sur IEEE 802.15.4 promet la pérennité et le développement de cette norme.

IEEE 802.15.4 souhaite également s'ouvrir aux nouveaux champs applicatifs que sont le transfert haut débit et la localisation indoor grâce à un nouvel amendement prévu pour 2022-2024 visant à introduire l'Ultra Wide Band (UWB) [30].

Bilan

À l'aide d'un échantillon de protocoles de l'IoT, nous avons constaté que lorsqu'un protocole est basé sur la norme IEEE 802.15.4, il est peu souvent possible, sachant sa couche physique, de prévoir le protocole de sa couche liaison de données.

Nous avons ensuite montré que les protocoles issus de la famille IEEE 802.15.4 constituent une part importante du marché de l'IoT, et que cette part est amenée, sinon à prendre de l'ampleur, tout au moins à perdurer dans le futur.

Compte tenu de ceci, on peut donc affirmer qu'étant donné la connaissance du protocole de la couche physique, il y aura de nombreux cas dans lesquels il ne sera pas possible de prévoir celui de la couche liaison de données.

En se basant sur les chiffres de prévisibilité concernant les autres couches dans notre échantillon, et d'après les perspectives d'IEEE 802.15.4, qui constitue une part non négligeable des cas d'imprévisibilité, on peut déduire que l'on aura la même imprévisibilité pour les couches réseau et application. Nous pouvons donc conclure que, dans le cas de l'IoT, il n'est pas toujours possible de prévoir le protocole d'une couche supérieure sachant celui de ses couches inférieures.

3 Problématique, famille d'adaptabilité choisie et état de l'art

3.1 Problématique

Pour les besoins d'une application, il est parfois nécessaire de faire cohabiter dans un même réseau plusieurs protocoles différents, pour leurs caractéristiques intrinsèques. Par exemple,

l'utilisation parallèle de WiFi pour les objets nécessitant des transferts haut débit, et de Zigbee pour ceux n'ayant besoin que d'une faible bande passante, le tout afin d'économiser au maximum l'énergie [35]. De la même manière, on peut parfaitement imaginer construire une pile protocolaire utilisant des couches issues de différents standards interconnectables afin de s'adapter aux besoins d'une application spécifique. Dans ce cas, comme on l'a vu précédemment, pouvoir démoduler et décoder le signal n'induit pas de savoir interpréter le flux binaire récupéré. On sort donc du cas classique, lorsque deux objets communicants connaissent parfaitement le ou les protocoles employés par l'autre, même si leurs implémentations respectives ne sont pas forcément les mêmes.

Dans ces conditions, un objet A ne peut donc pas communiquer avec un objet B si le ou les protocoles utilisés par B ne sont pas connus par A. Or, on se place dans la situation où un objet communicant est placé dans un environnement protocolaire lui étant inconnu, et il doit établir la communication avec les différents objets dans les réseaux à portée radio. La problématique est donc celle-ci : comment permettre à cet objet de communiquer dans (s'adapter à) ces différents réseaux sans connaître a priori les protocoles qu'ils utilisent ? Le scénario lié à cette problématique est illustrée par le schéma Figure 6.

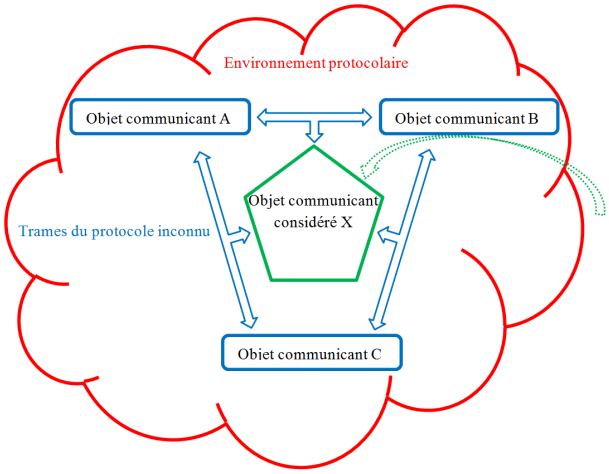


FIGURE 6 – Illustration du scénario de la problématique de la thèse

3.2 Choix du type d'adaptabilité

D'un point de vue macroscopique, deux familles de solutions sont envisageables :

— Adaptabilité par reconnaissance de protocoles, consistant à retrouver le protocole ou ses éléments dans une base de données [36] ;

— Adaptabilité par apprentissage de protocoles, consistant à reconstituer le protocole à partir des traces radio.

Nous allons étudier les avantages et les inconvénients de chacune de ces alternatives.

Adaptabilité par reconnaissance de protocole

Avantages :

Si le protocole auquel on cherche à s'adapter est compris dans la base de données de l'objet, il y a de très fortes chances que le protocole soit identifié [37], et donc que la communication soit assurée de parfaitement fonctionner.

Ensuite, la puissance de calcul nécessaire est généralement faible ou moyenne. Cela s'explique par le fait que les modèles choisis utilisent principalement des principes simples, comme le calcul de distance dans un espace précis entre le protocole à caractériser et les protocoles en mémoire [36], ou des algorithmes de Machine Learning peu gourmands comme les Support

Vector Machines (SVMs) [38].

De plus, les besoins en mémoire vive sont relativement limités car les échantillons utilisés pour l'identification sont d'une taille moins importante qu'en apprentissage, et le nombre d'attributs caractérisant les échantillons est plus limité (quelques dizaines) [36].

Enfin, les temps de traitement étant relativement faibles, la consommation en énergie l'est également, leur permettant d'être alimentés par une batterie de faible capacité.

Le faible temps de calcul associé à une taille d'échantillon nécessaire peu importante permet un rapide établissement de la connexion.

Inconvénients :
Si l'on considère le fait que de nouveaux protocoles, ou bien des variantes ou évolutions de ceux existants voient régulièrement le jour (exemple : voir l'historique de la norme IEEE 802.15.4 dans la section 2.2), il est très complexe de créer et maintenir à jour une plateforme adaptable universelle basée sur de la reconnaissance de protocole.

Une telle plateforme nécessite également un large espace de stockage afin que l'objet communicant ait en mémoire morte tous les protocoles auxquels il peut être confronté, même en factorisant au maximum les fonctions communes. Le format de prédilection utilisé dans les microcontrôleurs du marché de l'IoT est l'embedded flash memory (e-flash)[39], d'une capacité typique à l'heure actuelle autour de 512ko [40], tandis qu'une pile protocolaire complète typique en IoT tourne autour de 100ko [2][41]. On ne peut donc pas espérer stocker plus d'une dizaine de protocoles, ce qui ne constitue qu'une fraction du marché. Il est possible de prendre une mémoire de capacité plus importante, mais le coût unitaire augmente fortement, allant à l'encontre du principe de développement à grande échelle.

Adaptabilité par apprentissage de protocole

Avantages :
Une telle plateforme est théoriquement capable de s'adapter à n'importe quel protocole existant ou à venir, à partir du moment où celui-ci est conforme aux hypothèses sur lesquelles se fonde le modèle d'apprentissage.

L'occupation en mémoire morte est fortement réduite par rapport à une plateforme cherchant à effectuer de l'identification de protocole, car seul le modèle d'apprentissage est présent. Par exemple, pour des Reinforced Neural Networks (RNN), le poids du code compilé est de l'ordre de 100ko, et certains modèles spécialement développés pour l'IoT permettent de réduire ce poids en dessous de 10ko, sans perte notable de performance [42]. On peut donc raisonnablement supposer que n'importe quel modèle d'apprentissage, moyennant quelques adaptations éventuelles, puisse être adapté à un objet communicant.

Inconvénients :

2. Nous entendons par là que l'occupation mémoire du code compilé est d'environ 100ko. La cible sur laquelle l'évaluation a été faite est le Wismote, et l'OS utilisé Contiki

L'apprentissage de protocole étant plus complexe que la reconnaissance, et sa nature généraliste rendant impossible la prise en compte de tous les cas possibles (notamment parce que certains ne se produisent que rarement), la communication ne peut vraisemblablement pas être parfaitement fonctionnelle comme c'est le cas pour la reconnaissance, mais assurera plutôt un fonctionnement dans les cas d'usage 'normaux'.

Par ailleurs, la puissance de calcul nécessaire est généralement élevée voire très élevée, car les algorithmes de Machine Learning utilisés ont pour la plupart une complexité très importante. Par exemple, pour les mêmes RNN dont nous parlions [42], pour obtenir un temps d'apprentissage de l'ordre de l'heure, un CPU à 12 cœurs cadencés à 2.6GHz et un GPU offrant 12 TeraFLOPS ont été nécessaires. En parallèle, les objets communicants sont généralement dotés d'un processeur possédant un unique cœur faiblement cadencé (quelques centaines de MHz), et pas de GPU. On peut donc estimer que ce même apprentissage de quelques heures prendrait quelques mois sur un objet communicant, ce qui n'a pas de sens.

Les besoins en mémoire vive sont importants à cause de la taille de la base d'apprentissage. Il faut compter au minimum quelques milliers de messages [42], qui dans le contexte de l'IoT ne dépassent pas généralement les 255 octets ; l'espace mémoire nécessaire est donc de l'ordre de quelques Mo, ce qui dépasse assez largement les capacités typiques. Dans les faits, les objets communicants n'ont généralement qu'autour de 256ko de mémoire vive [40].

Les temps de calcul étant particulièrement longs et consommateurs, les objets doivent être équipés de batteries à forte capacité.

Les longs calculs, à effectuer sur de larges échantillons, donc longs à rassembler, demandent un temps relativement important avant de pouvoir initier la communication.

Bilan

Avec cette comparaison, nous avons pu constater que l'apprentissage de protocole offre une adaptabilité plus forte à son environnement, et que son noyau possède une plus faible empreinte en mémoire morte que la reconnaissance. Cependant, une qualité optimale de communication n'est pas assurée, la puissance de calcul, les besoins en mémoire vive, et la consommation en énergie sont importants, et le temps d'établissement de la communication est élevé, contrairement au cas de la reconnaissance de protocole.

Toutefois, l'adaptabilité est le critère sur lequel nous mettons ici le plus l'accent. Concernant les besoins en termes de performances de notre objet matériel, il est possible de se procurer des plateformes plus puissantes [43] (4 cœurs cadencés 1GHz, 512Mo de RAM), bien que cela se fasse au prix du coût unitaire de chaque objet. Pour la plateforme citée, il faut compter 4$ unité, ce qui, selon [44], est environ dix fois plus cher que le coût moyen actuel des objets communicants. Si l'on ajoute l'utilisation d'un framework optimisé pour l'IoT, combiné à un modèle d'apprentissage de complexité limitée, un apprentissage en quelques minutes devient envisageable, bien qu'au prix d'une partie des capacités d'apprentissage. De plus, la consommation en énergie associée aux calculs nécessiterait une batterie à forte capacité, mais faible encombrement, de type pile bouton, comme la BR3032 présentée dans [45], à un coût d'environ 2$ unité.

L'apprentissage nous semble donc au final le plus intéressant, c'est pourquoi nous le retenons dans le cadre de cette thèse et nous proposons un certain nombre d'éléments de réponse allant dans cette direction.

Une autre alternative pourrait consister à équiper l'algorithme d'apprentissage sur des coordinateurs de réseau, qui étendraient alors automatiquement tous les réseaux dont ils sont à portée. Ce genre de nœud réseau est généralement connecté à une source d'alimentation illimitée (courant secteur) et possède des performances bien plus élevées du fait qu'il est moins restreint en termes de prix unitaire, car largement inférieur en nombre aux nœuds terminaux.

3.3 État de l'art des systèmes d'établissement automatique de communication avec un protocole inconnu

D'après nos recherches, il n'existe actuellement aucune solution technique permettant à une entité communicante de s'adapter automatiquement à son environement protocolaire. Une solution technique s'en rapprochant consiste à juxtaposer plusieurs piles protocolaires [46]. Cette approche, appelée système multi-communications, comporte une pile protocolaire en deux parties : une première regroupant les couches hautes, c'est à dire généralement la couche applicative dans le cas de l'IoT, et une seconde regroupant les couches basses, à savoir physique, liaison de données, et réseau. La seconde partie est de manière pratique un conteneur incluant plusieurs piles basses distinctes, correspondant chacune à un protocole distinct. Un commutateur placé entre les deux parties de la pile globale permet de connecter à la partie supérieure la pile basse désirée, comme illustré sur la Figure 7. Toutefois, l'utilisation précise de chacune de ces piles et donc des protocoles est définie à l'avance, et un tel système peut par conséquent être vu comme utilisant un unique protocole composite, non adaptable à son environnement. Par ailleurs, insérer un nouveau nœud de télécommunication terminal dans ce réseau multi-communications nécessite de connaître exhaustivement l'ensemble des protocoles disponibles sur les coordinateurs du réseau. Sans compter que pour introduire un nouveau coordinateur, il est nécessaire de trouver sur le marché un produit équipé de l'ensemble de ces protocoles.

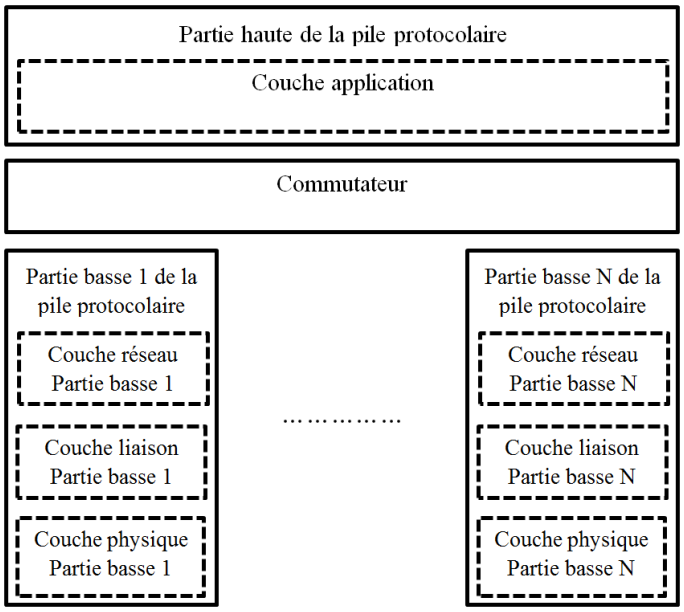


FIGURE 7 – Illustration du principe de juxtaposition de piles protocolaires dans l'IoT

Une autre solution technique consiste à utiliser une architecture à pile protocolaire reconfigurable, ce qui est rendu possible par la Radio Logicielle (RL) [47]. Cette approche consiste à décomposer les traitements des protocoles en fonctions et opérateurs communs, qui sont en-

suite assemblés pour former n'importe quel protocole. Cet assemblage, ou reconfiguration, peut être fait simplement à l'aide d'un ensemble de paramètres [48]. Ceci permet un changement rapide de pile protocolaire tout en minimisant l'empreinte mémoire, et un potentiel d'extensibilité élevé. Bien qu'originellement utilisé pour la couche physique des émetteurs-récepteurs classiques, son application peut être étendue aux couches supérieures et au domaine de l'IoT [49, 50]. Cependant, la RL se contente de permettre la reconfigurabilité, l'intelligence nécessaire pour le faire étant fournie par la Radio Intelligente. C'est donc de celle-ci que se rapproche l'objet de cette thèse. Au final la RL permettrait simplement la mise en œuvre d'un système utilisant cette intelligence.

D'après nos recherches, il n'existe pas de systèmes pouvant remplir tels quels le rôle de cette intelligence, cependant, un domaine semble consacré aux outils qui pourraient permettre le développement d'une telle intelligence : le Protocol Reverse Engineering (PRE), une famille de techniques visant à reconstruire les formats de trame et/ou la machine d'état d'un protocole inconnu cible en analysant les traces d'exécution et/ou les traces réseau. Il n'y a pas de procédure définie à proprement parler pour atteindre ce but, mais celle la plus commune [51] se décompose en six étapes principales :

1. Interception du trafic radio
2. Isolation des trames pertinentes du protocole inconnu
3. Identification des éléments caractéristiques du protocole inconnu
4. Regroupement des trames du protocole inconnu sur la base de leurs éléments caractéristiques
5. Alignement de séquences des trames du protocole inconnu au sein de chaque groupe
6. Reconstruction des formats de trames et/ou de la machine d'état du protocole inconnu

Nombre d'outils de PRE existent dans l'état de l'art, que ce soit pour l'analyse de protocoles binaires ou textuels, de couches hautes comme de couches basses, comme on peut le voir dans ces articles de synthèse de la littérature [52, 53]. Cependant, on peut noter que les protocoles textuels de couches hautes ont reçu plus d'attention que les protocoles binaires de couches basses, dans le cadre desquels nous nous plaçons dans cette thèse.

4 Démarche, orientation et organisation de la thèse

4.1 Démarche

Notre démarche de réflexion est la suivante : pour apprendre un langage lorsque l'on en connaît aucun, il faut d'abord l'entendre et en intégrer les structures, puis lui donner un sens en le reliant à nos a priori. On peut faire le parallèle avec le cas d'un enfant entendant ses parents parler : il apprend les mécanismes de la langue par mimétisme, sans pour autant en comprendre le sens, puis, progressivement, il relie les mots à des sensations physiologiques, puis des concepts, par essais-erreurs.

La démarche globale que nous envisageons pour permettre l'apprentissage d'un protocole est composée de deux principales étapes :

1. Un apprentissage dénué d'intelligence consistant à déterminer les formats de trames employés par les protocoles de la pile protocolaire inconnue, ainsi que leurs machines d'états. Ce point sera basé sur le PRE.
2. La compréhension des protocoles précédemment dégagés consistant à affecter une signification à chacun des états et formats à partir de connaissances possédées a priori.

4.2 Configuration d'étude de la thèse

Afin de définir un objectif en adéquation avec les attentes d'une thèse, nous avons procédé à un certain nombre de restrictions du cadre des recherches. Tout d'abord, nous limiterons notre travail au premier point de la démarche globale proposée. Ensuite nous ne nous concentrerons que sur l'apprentissage de formats de trames, et uniquement au niveau de la couche Liaison de Données. Lorsque des trames réelles seront utilisées, elles seront exclusivement issues de la norme 802.15.4 sur les LR-WPANs. Enfin, comme hypothèse simplificatrice, nous n'analyserons que des traces Liaison de Données comportant un seul format de trame.

Nous allons maintenant présenter la configuration d'étude exacte dans laquelle la thèse se place d'un point de vue pratique. On considère que les réseaux à portée de notre objet communiquent par voie radio, donc que nous n'avons accès qu'aux messages envoyés, sans information de directionalité. De plus, les protocoles des réseaux IoT à portée sont de type LR-WPAN, c'est à dire qu'ils sont conçus pour de nombreux nœuds avec des débits individuels faibles, et que la taille et complexité des messages est peu importante (peu de champs, taille totale ne dépassant généralement pas 256 octets). Enfin, nous considérons que les messages interceptés ont déjà été démodulés automatiquement par une couche physique dotée de reconnaissance de modulation [54]. Notre analyse portera donc uniquement sur la charge utile des trames physiques, et plus particulièrement leur en-tête.

4.3 Solution proposée

Le modèle d'apprentissage de formats de trames binaires de protocoles de couche Liaison de Données que nous proposons dans cette thèse est basé sur les réseaux Bayésiens. Le concept de réseaux Bayésiens a été introduit au début des années 1980 par Judea Pearl afin de pouvoir lier entre elles de façon graphique un ensemble de variables aléatoires d'un système [55]. Cette représentation prend la forme d'un graphe acyclique dont chaque sommet est une variable et dont chaque arc, orienté, matérialise une dépendance conditionnelle de la variable cible sur la variable source, comme illustré sur la Figure 8. Le nom Bayésien provient du fait que le théorème fondamental des probabilités conditionnelles est aussi appelé théorème de Bayes, du nom de son découvreur. Ce graphe permet alors, par l'application d'un algorithme ou procédure systématique, de déterminer toutes les probabilités du système à partir de ses probabilités marginales et conditionnelles connues. Cette procédure est appelée **inférence** et constitue la raison d'être des réseaux Bayésiens.

Les réseaux Bayésiens présentent un certain nombre d'avantages [56] :
— La nature graphique du modèle le rend plus facilement compréhensible et facilite la création de modèles et l'interprétation des résultats,
— Les calculs de probabilités complexes sont réalisés de manière très optimisée,

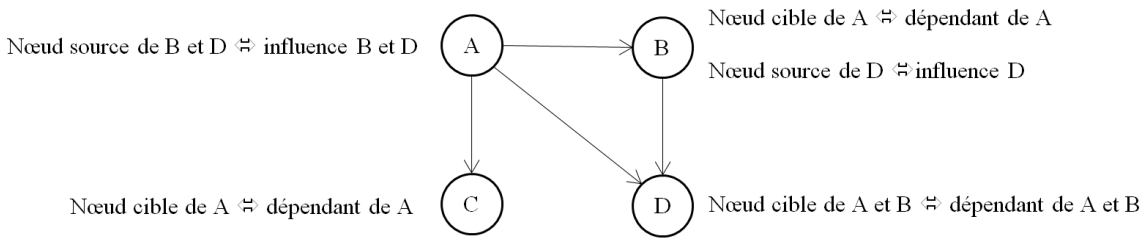


FIGURE 8 – Illustration d'un réseau Bayésien

— Il est relativement simple d'étendre un modèle existant, et cela ne change en rien la complexité de la procédure d'inférence.

Cependant, il existe aussi quelques inconvénients :

— Il est impossible de matérialiser des relations cycliques,
— La conception du réseau demande d'importants efforts,
— Un réseau donné est spécifique à une tâche et ne pourra pas s'adapter à une autre, comme c'est par exemple possible pour un réseau de neurones.

Dans le cas d'étude de notre thèse, les trames dont le format doit être appris ne présentent pas de relations cycliques dans les dépendances entre champs ; en effet, un champ ne dépend que des précédents. De plus, le modèle que nous cherchons à proposer n'a pas pour vocation d'être générique, mais spécifique à la tâche d'apprentissage de formats de trames de protocoles binaires de couches basses. Nous cherchons surtout un modèle facile à comprendre et à étendre (augmentation du nombre de variables utilisées et création de nouvelles relations probabilistes) sans explosion de sa complexité. Les réseaux Bayésiens comportent donc nombre d'avantages pour nous, et peu d'inconvénients.

4.4 Plan du manuscrit

Suite à cette introduction, dans le **chapitre 1**, nous nous intéresserons à l'état de l'art en matière de PRE. Les détails exacts du fonctionnement global des outils de la littérature étant généralement omis dans les articles les présentant, nous ne nous sommes intéressés qu'à une seule de leur composante : la méthode d'identification de séquences. Nous avons trouvé trois articles présentant des explications suffisamment détaillées de la technique employée pour réaliser cette opération pour pouvoir les implémenter et simuler nous-mêmes. Les trois algorithmes retenus sont : Variance Distribution of the Variances (VDV) [57], un Aho-Corasick modifié (AC) [58], et Latent Dirichlet Allocation (LDA) [59]. Dans ce chapitre, nous présenterons la théorie qui sous-tend chacune de ces techniques, ainsi que la façon dont ils ont été adaptés au contexte de la thèse pour les simulations, et enfin les variables dont l'influence sera étudiée dans le chapitre suivant.

Dans le **chapitre 2**, nous comparerons les performances des trois techniques précédemment mentionnées, ce qui à notre connaissance n'a jamais été fait, en tous cas pas à ce niveau de détail. Nous présenterons d'abord rapidement le simulateur que nous avons construit pour effectuer les simulations. Nous détaillerons ensuite les diverses métriques employées pour évaluer les performances des trois techniques : la précision, la couverture (recall), ainsi que le score F, auxquelles le ratio de détection des champs, une nouvelle métrique que nous avons

conçue, a été adjoint. Nous discuterons ensuite les résultats obtenus et conclurons que la technique LDA est la plus performante des trois, ce qui nous orientera vers le domaine des réseaux Bayésiens, dont le LDA est issu.

Dans le **chapitre 3**, nous présenterons la contribution principale de cette thèse, à savoir le modèle d'apprentissage de format de trame, BaNet3F. Dans ce but, nous commencerons par présenter succinctement la théorie des réseaux Bayésiens et des réseaux Bayésiens dynamiques, et plus précisément leurs principes, représentations, et comment effectuer la procédure d'inférence, ainsi que celle d'apprentissage. Pour de plus amples détails, des exemples, et les diverses notions connexes, nous inviterons le lecteur à lire les annexes. Nous présenterons ensuite le principe théorique du modèle proposé, BaNet3F, ainsi que le réseau Bayésien utilisé en son cœur. Nous introduirons également une adaptation de l'algorithme de Viterbi aux réseaux Bayésiens, AMBV, permettant de trouver l'état le plus probable d'un réseau Bayésien quelconque de façon relativement optimisée.

Dans le **chapitre 4**, nous présenterons d'abord le générateur de traces binaires employé pour évaluer les performances de notre modèle. Ensuite, nous montrerons que notre modèle atteint des performances correctes, variant entre 50% et 100% selon la complexité de la trace analysée. Nous verrons ensuite que les performances sur traces réelles sont moins bonnes, ne dépassant guère les 50% dans les cas les plus favorables, et nous expliquerons pourquoi.

Enfin, dans le **chapitre 5**, nous comparerons les performances de notre modèle à celles de deux techniques de l'état de l'art : LDA, déjà étudié, et le modèle de Cai et al. [60], basé sur les chaînes de semi-Markov cachées, que nous avons pris comme point de départ pour le développement de notre modèle. Nous montrerons que les performances de notre modèle sont nettement supérieures à celles issues de l'état de l'art lorsque la trace est peu complexe, mais que cet écart se réduit lorsque la complexité de la trace augmente.

Chapitre 1

Étude des algorithmes VDV, AC et LDA

Dans l'optique de procéder à l'apprentissage d'un protocole, il nous a semblé intéressant d'étudier la littérature dans le domaine du Protocol Reverse Engineering (PRE). Le PRE [52] consiste à étudier les traces réseaux ou d'exécution d'un protocole inconnu pour en déterminer le fonctionnement interne. La procédure la plus répandue pour y parvenir se décompose en six étapes principales, comme schématisé sur la Figure 1.1 :

1. Interception du trafic radio
2. Isolation des trames pertinentes du protocole inconnu
3. Identification des éléments caractéristiques (séquences binaires remarquables dans notre cas) du protocole inconnu
4. Regroupement des trames du protocole inconnu par familles sur la base de leurs éléments caractéristiques
5. Alignement de séquences des trames du protocole inconnu au sein de chaque groupe
6. Reconstruction des formats de trames et/ou de la machine d'état du protocole inconnu

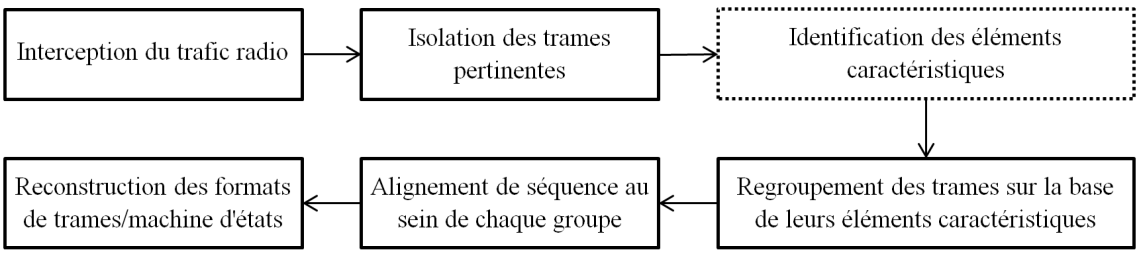


FIGURE 1.1 – Procédure de PRE la plus répandue

Afin de comprendre en détail le fonctionnement de ces systèmes, et de tester leurs performances, nous avons sélectionné quelques algorithmes utilisés dans la littérature pour effectuer de l'identification de séquences remarquables.

Dans ce chapitre nous présenterons les concepts fondateurs des différents algorithmes ainsi que les éléments essentiels de leur implémentation. Il existe une multitude de techniques pour

effectuer l'identification de séquences, mais nous n'en exposerons que trois, classées par ordre de complexité croissante :

— Variance of the Distribution of Variances (VDV)
— Aho-Corasick (AC)
— Latent Dirichlet Allocation (LDA)

Pour chacun de ces algorithmes, nous présenterons le principe général, le fonctionnement détaillé, puis les adaptations nécessaires pour l'appliquer à la configuration d'étude de la thèse.

1.1 Objectifs

L'identification de séquences a pour objectif d'extraire les séquences caractéristiques d'un message, et donc de réduire la quantité d'informations nécessaire pour le caractériser. Cela est réalisé en identifiant les zones remarquables de la trame DLL, c'est à dire généralement en repérant les séquences récurrentes et leurs positions. Lorsqu'une telle séquence est repérée, il y a de fortes chances pour qu'elle corresponde à un mot-clé du protocole inconnu.

Afin d'illustrer les propos précédents, considérons un exemple concret. Ci-dessous est représenté (Figure 1.2) un des formats de trame du protocole Enocéan. Les champs hachurés sont ceux qui doivent être identifiés dans ce procédé. En effet, tous les autres champs sont variables et hautement dépendant du contexte particulier d'une communication donnée.

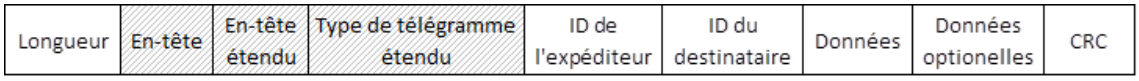


FIGURE 1.2 – Trame DLL Enocéan pour une charge utile supérieure à six octets

1.2 Algorithme Variance of the Distribution of Variances (VDV)

Cet algorithme cherche à identifier de manière statistique dans une population donnée les zones d'un individu (au sens général) offrant la plus faible variabilité. Cette approche a été proposée par A. Trifilò et al [57] dans leurs travaux sur le PRE.

1.2.1 Principe

L'algorithme VDV considère une population formée de groupes d'individus. Ces derniers sont composés d'éléments pouvant prendre différentes valeurs numériques. La structuration hiérarchique de la population est illustrée sur le schéma de la Figure 1.3.

L'algorithme se décompose en cinq étapes, comme illustré sur le schéma de la Figure 1.4 :

— Le calcul de la valeur moyenne de chaque élément des individus au sein de chaque groupe
— Le calcul de la variance de chaque élément des individus au sein de chaque groupe
— le calcul de la moyenne de ces variances
— Le calcul de la variance de ces variances

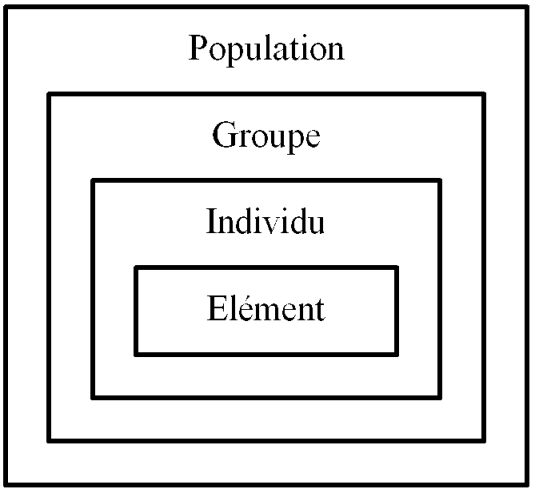


FIGURE 1.3 – Structuration hiérarchique d'une population au sens de l'algorithme VDV

— Le filtrage des éléments présentant une variance des variances inférieure à un seuil

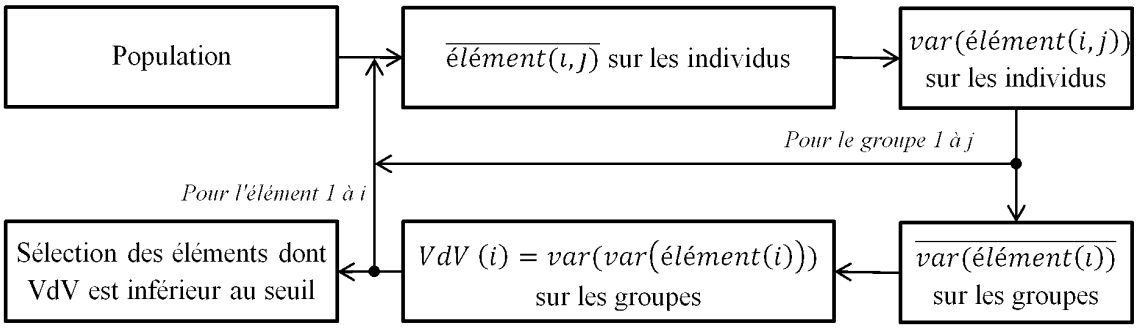


FIGURE 1.4 – Enchaînement des opérations dans l'algorithme VDV

$\bar{x}$ représente la moyenne de x, et $var(x)$ la variance de x.

1.2.2 Algorithme

L'algorithme VDV est présenté de façon simplifiée sous forme de pseudo-code dans l'Algorithme 1.

1.2.3 Application à la configuration d'étude de la thèse

L'algorithme utilisé dans la configuration d'étude de la thèse dérive de celui présenté dans les sections 1.2.1 et 1.2.2, auquel nous avons apporté les modifications suivantes :

— Les groupes d'individus considérés dans l'algorithme VDV précédent sont remplacés par des flux constitués de trames DLL à analyser, et l'unité de base qu'était l'élément est remplacée par le "jeton", constitué de n bits. Cette équivalence est résumée dans la Table 1.1. Une trame DLL est donc constituée d'une séquence de jetons caractérisés par leur position.

Algorithme 1 : Algorithme VDV

Données : données, seuil de filtrage
Résultat : Positions des éléments invariants

```
pour chaque groupe faire
    pour chaque élément faire
        On calcule la valeur moyenne sur les individus pour l'élément considéré au sein du groupe considéré;
    fin
fin
pour chaque groupe faire
    pour chaque élément faire
        On calcule la variance des valeur sur les individus pour l'élément considéré au sein du groupe considéré;
    fin
fin
pour chaque élément faire
    On calcule la moyenne des variances sur les groupes pour l'élément considéré;
fin
pour chaque élément faire
    On calcule la variance des variances sur les groupes pour l'élément considéré;
    si variance des variances de l'élément < seuil de filtrage alors
        On extrait l'élément;
    fin
fin
```

— Afin de pouvoir détecter des champs à n'importe quelle position, tous les jetons de n bits consécutifs qu'il est possible de créer à partir d'une trame donnée sont créés.
— Pour permettre la détection de champs de longueurs différentes, l'algorithme est exécuté plusieurs fois, pour un ensemble de valeurs de n, et toutes les séquences uniques extraites sur l'ensemble des itérations sont conservées pour analyse des performances.
— Le seuil de filtrage utilisé ne représente pas une valeur absolue, mais est proportionnel à la moyenne des variances des variances des jetons. Le coefficient de proportionnalité est appelé coefficient du seuil de filtrage de VDV (CSF_{VDV}), et la relation le liant au seuil de filtrage de VDV SF_{VDV} est la suivante :

$$SF_{VDV} = \overline{VdV(i)} \times CSF_{VDV}, \tag{1.1}$$

où $VdV(i)$ représente la variance des variances du jeton i.
— Les trames à analyser ayant été récupérées dans un contexte radio, il n'est pas possible d'identifier clairement des flux de messages, ceux-ci seront donc créés artificiellement par affectation aléatoire des trames aux flux selon une loi uniforme discrète.

Les différentes étapes du processus sont illustrées sur le schéma de la Figure 1.5.

Terme général employé dans l'exposition du principe	Terme équivalent employé dans le cas de la configuration d'étude de la thèse
Population	Trace liaison de données
Groupe	Flux
Individu	Trame
Élément	Jeton

TABLE 1.1 – Récapitulatif des termes équivalents

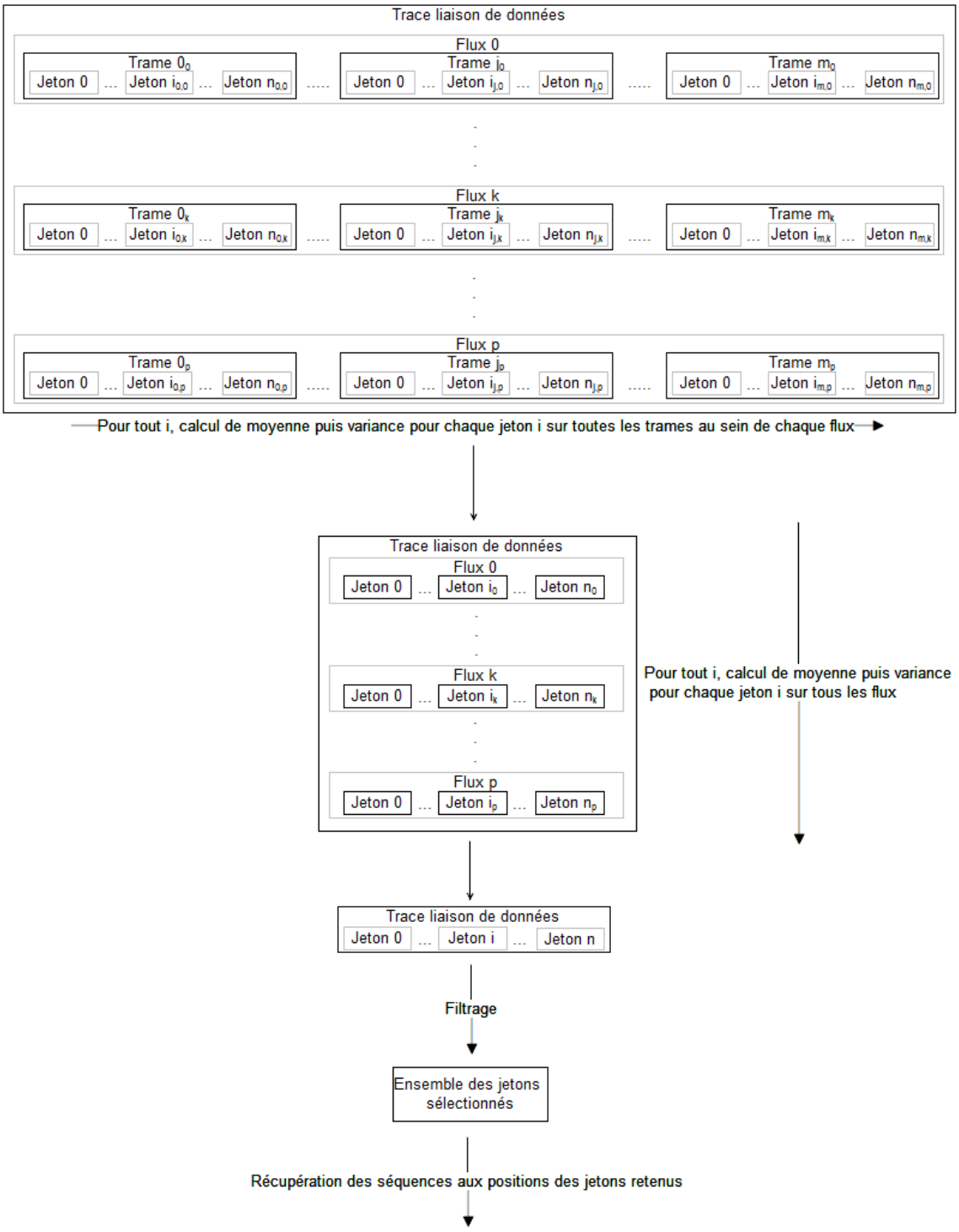


FIGURE 1.5 – Schéma du principe de fonctionnement de l'algorithme VDV

Les paramètres dont nous étudierons l'influence sur les performances de l'algorithme seront les suivants :

— Longueur maximale des séquences recherchées (longueur des jetons)
— Valeur du seuil de filtrage
— Nombre de flux générés

1.3 Algorithme d'Aho-Corasick (AC)

Cet algorithme a été conçu par A. Aho et M. Corasick dans le but d'identifier une chaîne de caractères dans un texte [61]. Cependant, son utilisation peut être étendue à l'identification de n'importe quel motif constitué d'une séquence d'éléments, chacun d'entre-eux prenant sa valeur dans un ensemble discret fini. La recherche est alors effectuée au sein d'une suite de ces éléments de longueur supérieure ou égale à celle du motif recherché. Cette approche a été proposée par Y. Wang et al.[58] dans leurs travaux sur le PRE.

1.3.1 Principe

L'algorithme AC, dans une optique d'efficacité en temps de calcul, est basé sur une machine d'état.

Les états de cette machine sont caractérisés par le caractère devant être lu pour accéder à cet état particulier. Cet automate est régi par trois fonctions :

— Une fonction de transition g, déterminant l'état suivant à emprunter en fonction de l'état actuel et du caractère lu lorsque celui-ci est pertinent par rapport à l'arbre des mots recherchés
— Une fonction d'échec f, déterminant l'état suivant à emprunter en fonction de l'état actuel lorsque le caractère lu n'est pas pertinent par rapport à l'arbre des mots recherchés
— Une fonction d'activité o, déterminant le ou les mots se terminant en fonction de l'état actuel

L'algorithme se déroule en deux temps :

— La construction de la machine d'état à partir des mots à rechercher dans le texte
— Le balayage par l'automate de chaque caractère du texte. À chaque caractère lu, les mots trouvés peuvent être extraits à l'aide de la fonction o.

Sur la Figure 1.6, un schéma de la machine d'état de l'algorithme est présenté pour un cas concret. Les mots recherchés sont : pas, peu, as, pass.

Chaque cercle représente un état. La lettre inscrite sur chaque état est celle nécessaire pour y accéder. L'état vide correspond à la racine de l'arbre. Les traits épais rectilignes matérialisent les transitions gérées par la fonction g. Ils sont parcourus exclusivement en descendant l'arbre. Les traits fins courbes matérialisent les liaisons gérées par la fonction f. Ils sont parcourus exclusivement en remontant l'arbre. Les cercles plus épais autour de certains états signifient qu'un ou plusieurs mots se finissent à cet endroit. Ces derniers sont indiqués à côté de l'état correspondant.

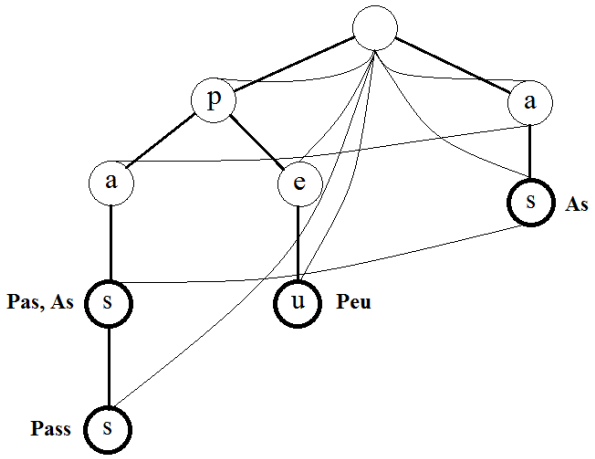


FIGURE 1.6 – Exemple de représentation conceptuelle d'une machine d'état de l'algorithme AC

1.3.2 Algorithme

L'algorithme AC est présenté de façon simplifiée sous forme de pseudo-code dans l'Algorithme 2.

1.3.3 Application à la configuration d'étude de la thèse

L'algorithme utilisé dans la configuration d'étude de la thèse dérive de celui présenté dans les sections 1.3.1 et 1.3.2, auquel nous avons apporté les modifications suivantes :

— Le texte considéré dans l'algorithme AC est remplacé par la trace liaison de données à analyser, et l'unité de base qu'était le caractère est remplacée par le bit. De plus, les mots recherchés deviennent toutes les séquences possibles de n bits. Cette équivalence est résumée dans la Table 1.2.

— Un compteur d'occurrence des séquences a été rajouté, et celles au-dessus du seuil suivant sont sélectionnées :

$$SF_{AC} = \frac{n - Len + 1}{2^{Len}} \times CSF_{AC}, \tag{1.2}$$

avec n le nombre de bits de la trace, Len la longueur de la séquence recherchée, et CSF_{AC} un coefficient ajustable par l'utilisateur. Ceci est effectué dans le but de ne retenir que les séquences suffisamment fréquentes.

— Fusion des séquences ayant un niveau de similarité Sim supérieur au seuil de fusion. Par fusion, on entend que, dans un groupe de séquences similaires, on choisit l'une comme représentante des autres, grâce à une classification arborescente hiérarchique. La similarité est définie par l'expression 1.3 suivante :

$$Similarité = \frac{longueur(X,Y) - ed(X,Y)}{longueur(X,Y)}, \tag{1.3}$$

avec X et Y deux séquences,

$$longueur(X,Y) = \frac{longueur(X) + longueur(Y)}{2}, \tag{1.4}$$

Algorithme 2 : Algorithme AC

Données : texte,mot
Résultat : Positions des mots recherchés

```
Tous les éléments de o sont initialisés à 0;
Tous les éléments de g et f sont initialisés à -1;
pour tous les mots faire
    Générer la branche de l'arbre correspondante avec g;
    Associer le dernier état de la branche à la chaîne de caractères achevée avec o;
fin
pour chaque caractère du dictionnaire faire
    Renvoyer la racine vers elle-même avec g pour les caractères ne débutant pas de
     mot;
    pour chaque état associé à ce caractère faire
        Insérer dans une queue l'état s'il correspond à un début de mot;
        si l'état n'a aucun état suivant d'après g alors
            si aucun autre état ne correspond à ce caractère alors
                Renvoyer l'état vers la racine avec f;
            fin
            Renvoyer avec f l'état à l'un des autres états correspondant à ce caractère;
            Répéter l'opération précédente jusqu'à ce que tous les états correspondant
             à ce caractère forment une chaîne, puis finalement lier le dernier état de la
             chaîne à la racine;
        fin
    fin
fin
tant que la queue n'est pas vide faire
    Prendre le premier état de la queue;
    pour chaque caractère du dictionnaire faire
        si il y a un état correspondant sur une branche alors
            Trouver l'état de plus grande profondeur dont la chaîne de caractères
             associée est une sous-chaîne de celle associée à l'état actuel;
            Associer à l'état trouvé, dans o, la chaîne de caractères achevée à l'état
             suivant de la branche;
            Introduire l'état suivant dans la queue;
        fin
    fin
fin
Démarrer à la racine;
pour chaque caractère du texte faire
    tant que il n'y a pas d'état suivant avec g faire
        Utiliser f pour définir l'état suivant;
    fin
    Mettre à jour l'état actuel avec g;
    si un mot se finit dans l'état actuel alors
        Le récupérer avec sa position de fin, et en déduire le début;
    fin
fin
```

et ed(X,Y) la distance minimale d'édition entre X et Y. Il s'agit du nombre minimal d'opérations (ajout, suppression, modification d'un bit) à effectuer sur une séquence pour obtenir l'autre. Toutes les séquences faisant la même longueur, longueur(X,Y) = longueur(X) = longueur(Y).

— Afin de pouvoir détecter toutes les longueurs de séquences remarquables, l'algorithme balaye toutes les longueurs de séquences inférieures ou égales à la valeur fixée en paramètre.

Terme général employé dans l'exposition du principe	**Terme équivalent employé dans le cas de la configuration d'étude de la thèse**
Texte	Trace liaison de données
Chaîne de caractères	Séquence de bits
Caractère	Bit

TABLE 1.2 – Récapitulatif des termes équivalents

L'algorithme final utilisé est illustré sur le schéma de la Figure 1.7.

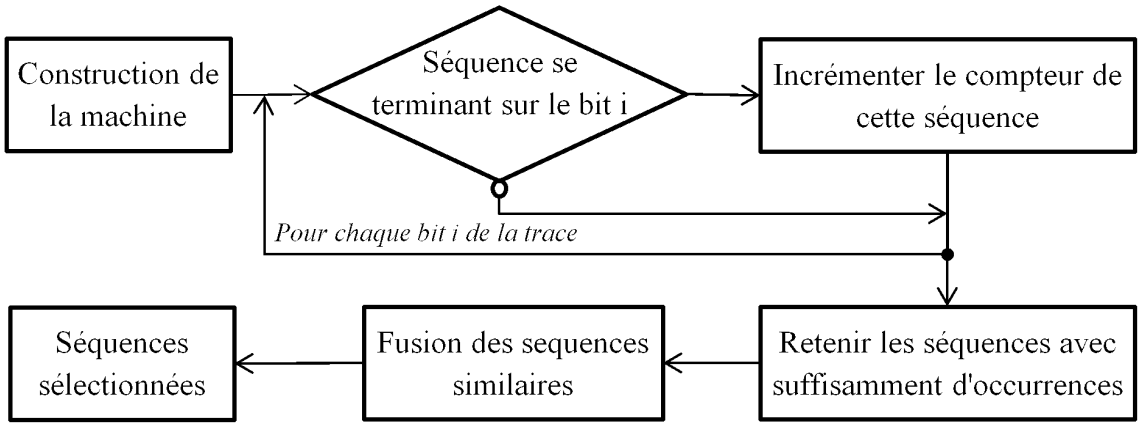


FIGURE 1.7 – Enchaînement des opérations dans l'algorithme AC

Les paramètres dont nous étudierons l'influence sur les performances de l'algorithme seront les suivants :

— Longueur maximale des séquences recherchées
— Valeur du seuil de filtrage
— Valeur du seuil de fusion

1.4 Algorithme Latent Dirichlet Allocation (LDA)

Cette technique [62] est issue du domaine de l'Information Retrival (IR), visant à modéliser un texte mathématiquement, afin d'en extraire le sens. Elle a été conçue dans le but d'identifier des sujets à partir d'un corpus de documents, et d'y associer des termes issus d'un dictionnaire. Cependant, son utilisation peut-être étendue au regroupement de séquences d'éléments uniques

prenant leurs valeurs dans un ensemble discret fini, à partir d'un ensemble de données. Cette approche a été proposée par Y. Wang et al [59] dans leur modèle de PRE ProDecoder.

1.4.1 Choix de la loi de Dirichlet

Le concepteur de LDA a choisi d'utiliser la loi de Dirichlet (cf Annexe 1) comme distribution a priori des paramètres des lois de probabilités des mots par sujets et des sujets par documents, car elle présente des propriétés mathémathiques simplifiant l'inférence des variables cachées et l'estimation des paramètres optimaux. Elle est notamment conjuguée avec la loi multinomiale, c'est à dire que si l'on considère une variable dont la loi de probabilité a priori suit une loi de Dirichlet, alors sa loi de probabilité a posteriori suit une loi multinomiale. La loi multinomiale étant destinée à de la catégorisation, ce qui est le but de LDA, la loi de Dirichlet est la plus adaptée comme loi a priori. Cette loi permet, à partir d'un vecteur de paramètres de concentration, de générer un vecteur de probabilité caractérisant une loi multinomiale. Ces paramètres ont des valeurs variant de 0 à $+\infty$. "0" signifie que les probabilités du vecteur de probabilité généré seront 0 ou 1 (avant normalisation), ce qui correspond à une loi binaire. "$+\infty$" signifie que les probabilités seront toutes égales à 0,5 (avant normalisation), ce qui correspond à une loi uniforme. "1" signifie que l'a priori est en fait qu'il n'y a pas d'a priori, c'est-à-dire que les probabilités peuvent prendre n'importe quelle valeur entre 0 et 1 avec une probabilité égale.

1.4.2 Modèle génératif

Le LDA est avant tout un modèle génératif issu des réseaux bayésiens (cf section 3.1) matérialisant l'hypothèse régissant la création d'un corpus. Le modèle LDA est dit latent car il possède un ensemble de variables cachées : les sujets auxquels sont rattachés les mots. L'algorithme concrètement implémenté en découle ensuite par application d'une méthode d'apprentissage.

Nous introduisons tout d'abord les différentes notions et paramètres suivants, nécessaires à la compréhension du modèle génératif et de l'inférence basée dessus :

- Un mot w est un élément issu du dictionnaire v rassemblant le vocabulaire connu
- Un document m est un ensemble de mots w, modélisé par un vecteur
- Un corpus W est un ensemble de documents m, modélisé par un vecteur
- V représente l'ensemble du vocabulaire ou la taille du dictionnaire v des termes disponibles
- K représente l'ensemble des sujets recherchés ou leur nombre
- M représente l'ensemble des documents dans le corpus ou leur nombre
- α est le paramètre de Dirichlet a priori de la distribution des sujets par document
- β est le paramètre de Dirichlet a priori de la distribution des mots par sujet
- ξ est le paramètre de Poisson de la loi régissant le nombre de mots de chaque document
- $\vec{\theta_m}$ est le vecteur caractérisant la distribution des sujets pour le document m. On note $\Theta = \left\{\vec{\theta_m}\right\}_{m=1}^{M}$, la matrice M $\times$ K caractérisant la distribution des sujets en fonction des documents.
- $\vec{\phi_k}$ est le vecteur caractérisant la distribution des termes de v pour le sujet k. On note $\Phi = \left\{\vec{\phi_k}\right\}_{k=1}^{K}$, la matrice K $\times$ V caractérisant la distribution des termes en fonction

des sujets.
— N_m représente le nombre de mots du document m.
— $w_{m,n}$ représente le $n^{ième}$ mot du document m. Le vecteur $\vec{w_m}$ représente les mots du document m. Le vecteur de vecteurs M $\times$ N_m $\vec{w}$, représente les mots du corpus.
— $z_{m,n}$ représente le sujet associé au $n^{ième}$ mot du document m. Le vecteur $\vec{z_m}$ représente les sujets respectivement affectés à chaque mot du document m. Le vecteur de vecteurs M $\times$ N $\vec{z}$, représente les sujets respectivement affectés à chaque mot du corpus.

Considérons par exemple un corpus de trois documents :
— Document 1 : Ceci est un exemple
— Document 2 : Les documents n'ont pas tous le même nombre de mots
— Document 3 : Les matrices sont des vecteurs de vecteurs

Sous forme matricielle, et pour 3 sujets, on pourrait avoir les matrices suivantes :

— Pour $\vec{w}$:
$$\begin{pmatrix} \begin{pmatrix} Ceci \\ est \\ un \\ exemple \end{pmatrix} & \begin{pmatrix} Les \\ documents \\ n'ont \\ pas \\ tous \\ le \\ même \\ nombre \\ de \\ mots \end{pmatrix} & \begin{pmatrix} Les \\ matrices \\ sont \\ des \\ vecteurs \\ de \\ vecteurs \end{pmatrix} \end{pmatrix}$$

— Pour $\vec{z}$, par exemple :
$$\begin{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \end{pmatrix} & \begin{pmatrix} 3 \\ 2 \\ 2 \\ 1 \\ 2 \\ 2 \\ 1 \\ 3 \\ 2 \\ 3 \end{pmatrix} & \begin{pmatrix} 3 \\ 2 \\ 1 \\ 3 \\ 1 \\ 2 \\ 1 \end{pmatrix} \end{pmatrix}$$

D'après les matrices précédentes, on sait donc que le mot "Ceci" est associé au sujet 1, le mot "un" est associé au sujet 2, et les deux instances du mot "Les" sont associées au sujet 3.

Afin de fixer les idées, voici un exemple de matrice Φ (la matrice Θ possède la même structure) correspondant aux matrices $\vec{w}$ et $\vec{z}$ précédemment présentées. Les indices des mots sont définis par rapport à leur première apparition dans le corpus. Φ comporte 3 sujets recherchés et 18 termes dans le dictionnaire. Les colonnes matérialisent les sujets, et les lignes les termes. Chaque élément de la matrice représente la probabilité d'obtenir le terme de la ligne dans laquelle il se trouve sachant le sujet de la colonne où il est. On a donc une somme des probabilités de chaque colonne égale à 1, car l'on doit forcément tirer un terme pour chaque

sujet donné.

$$\Phi = \begin{pmatrix} 0.125 & 0 & 0 \\ 0.125 & 0 & 0 \\ 0 & 0.125 & 0 \\ 0.125 & 0 & 0 \\ 0 & 0 & 0.4 \\ 0 & 0.125 & 0 \\ 0 & 0.125 & 0 \\ 0.125 & 0 & 0 \\ 0 & 0.125 & 0 \\ 0 & 0.125 & 0 \\ 0.125 & 0 & 0 \\ 0 & 0 & 0.2 \\ 0 & 0.25 & 0 \\ 0 & 0 & 0.2 \\ 0 & 0.125 & 0 \\ 0.125 & 0 & 0 \\ 0 & 0 & 0.2 \\ 0.25 & 0 & 0 \end{pmatrix}$$

Dans le cadre de LDA, on considère qu'un corpus est un ensemble de documents. Chacun de ces documents est composé d'un nombre aléatoire de mots, tiré selon une loi de Poisson de paramètre ξ. Ces mots prennent chacun une valeur parmi l'ensemble de celles d'un dictionnaire de termes.

On commence par générer aléatoirement les matrices Φ et Θ à l'aide d'une loi de Dirichlet de paramètre respectivement β et α. Ces matrices représentent un ensemble de vecteurs de probabilités multinomiales.

Pour chaque mot à générer de chaque document à créer, dans un premier temps, on tire aléatoirement à l'aide d'une loi multinomiale paramétrée par Φ le sujet auquel il appartiendra, sachant le document dans lequel il est. On tire ensuite aléatoirement à l'aide d'une loi multinomiale paramétrée par Θ le terme choisi, sachant le sujet auquel il appartient, tiré précédemment.

Le processus de génération des documents selon LDA est décrit dans l'Algorithme 3 et résumé sur la Figure 1.8. Les flèches représentent les liens de causalité (la loi de probabilité) liant les variables aléatoires, représentées par les cercles. Les zones délimitées par les rectangles en pointillés matérialisent les différents ensembles d'éléments formant le corpus :

— Les sujets k
— Les documents m
— Les positions de mots n

Les cercles pointillés correspondent aux variables cachées du modèle (dites aussi latentes), par opposition aux autres variables étant visibles. La variable hachurée est la variable finale

générée par le modèle : les mots présents dans le corpus.

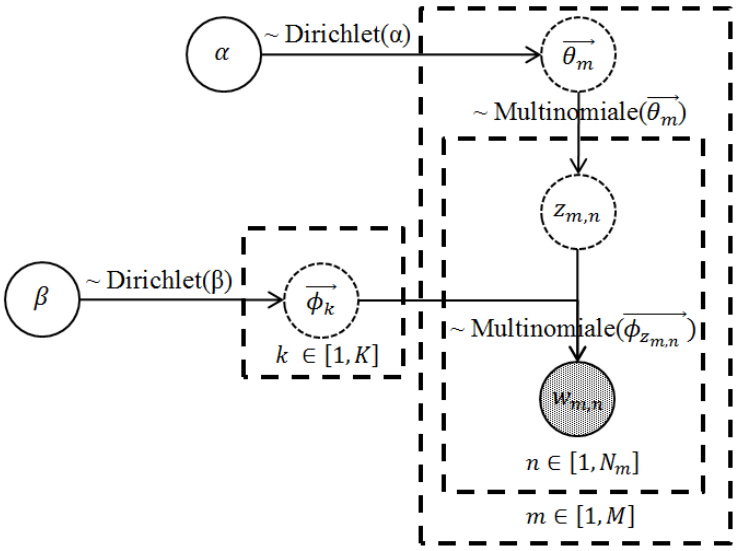


FIGURE 1.8 – Représentation par plaques du modèle génératif de LDA

Il est à noter que, dans le cas général, α et β sont des matrices, et ξ est un vecteur, mais dans le cas de LDA, dans l'Algorithme 3, nous les utilisons comme des scalaires, car nous considérons toutes les composantes au sein d'un même vecteur ou matrice comme égales.

On génère $\vec{\beta}$, $\vec{\alpha}$ et $\vec{\xi}$ en clonant, respectivement, les valeurs de β, α, et ξ autant de fois que nécessaire, soit : $\vec{\beta} = \{\beta\}_{k=1v=1}^{k=K,v=V}$, $\vec{\alpha} = \{\alpha\}_{m=1,k=1}^{m=M,k=K}$, et $\vec{\xi} = \{\xi\}_{m=1}^{m=M}$. De plus, ξ, n'a de sens que pour le modèle génératif, il ne sera plus utilisé lors de l'adaptation à un cas concret.

1.4.3 Inférence de Θ et Φ

L'objectif de l'algorithme LDA est d'inférer les distributions des mots par sujet et des sujets par document, c'est à dire les matrices Φ et Θ, à partir du corpus de documents.

Sachant les paramètres α et β, pour un document, la probabilité conjointe d'obtenir toutes les variables observables $(\vec{w_m})$ et cachées $(\vec{z_m}, \vec{\theta_m}, \Phi)$ est donnée par :

$$p(\vec{w_m}, \vec{z_m}, \vec{\theta_m}, \Phi|\alpha, \beta) = \prod_{n=1}^{N_m} p(w_{m,n}|\vec{\phi_{z_{m,n}}}) p(z_{m,n}|\vec{\theta_m}) p(\vec{\theta_m}|\alpha) p(\Phi|\beta). \tag{1.5}$$

$\vec{\phi_{z_{m,n}}}$ représente le vecteur de probabilité de la matrice Φ associé à $z_{m,n}$, le sujet associé au mot à la position n du document m.

Afin d'obtenir la distribution conjointe de $\vec{w_m}$ et $\vec{z_m}$, on intègre sur $\vec{\theta_m}$ et Φ :

Algorithme 3 : Modèle génératif de LDA

Données : M,K,v,α,β,ξ
Résultat : $\vec{w}$

pour *k = 1 à K* **faire**
  On génère $\vec{\phi_k}$ suivant une loi de Dirichlet de paramètre $\vec{\beta}$: $\vec{\phi_k} \sim$ Dirichlet($\vec{\beta}$);
fin
pour *m = 1 à M* **faire**
  On génère $\vec{\theta_m}$ suivant une loi de Dirichlet de paramètre $\vec{\alpha}$: $\vec{\theta_m} \sim$ Dirichlet($\vec{\alpha}$);
  On tire aléatoirement un nombre de mots suivant une loi de Poisson de paramètre ξ : $N_m \sim$ Poisson(ξ);
  pour *n = 1 à N_m* **faire**
    On tire aléatoirement un sujet $z_{m,n}$ selon une loi multinomiale de paramètre $\vec{\theta_m}$: $z_{m,n} \sim$ multinomiale($\vec{\theta_m}$);
    On tire aléatoirement un mot $w_{m,n}$ dans v selon une loi multinomiale de paramètre $\vec{\phi_{z_{m,n}}}$: $w_{m,n} \sim$ multinomiale($\vec{\phi_{z_{m,n}}}$);
  fin
fin

$$p(\vec{w_m}, \vec{z_m}|\alpha, \beta) = \int\int p(\vec{\theta_m}|\alpha)p(\Phi|\beta)\prod_{n=1}^{N_m} p(w_{m,n}|\vec{\phi_{z_{m,n}}})p(z_{m,n}|\vec{\theta_m})d\Phi d\vec{\theta_m}. \tag{1.6}$$

Afin d'obtenir la distribution marginale de $\vec{w_m}$, on somme $p(\vec{w_m}, \vec{z_m}|\alpha, \beta)$ sur toutes les valeurs de $z_{m,n}$, à savoir tous les sujets K :

$$\begin{aligned} p(\vec{w_m}|\alpha, \beta) &= \int\int p(\vec{\theta_m}|\alpha)p(\Phi|\beta)\prod_{n=1}^{N_m}\sum_{z_{m,n}} p(w_{m,n}|\vec{\phi_{z_{m,n}}})p(z_{m,n}|\vec{\theta_m})d\Phi d\vec{\theta_m} \\ &= \int\int p(\vec{\theta_m}|\alpha)p(\Phi|\beta)\prod_{n=1}^{N_m} p(w_{m,n}|\vec{\theta_m}, \Phi)d\Phi d\vec{\theta_m}. \end{aligned} \tag{1.7}$$

La probabilité conjointe de tous les mots du corpus et leur sujets associés est finalement déterminée par :

$$p(\vec{w}, \vec{z}|\alpha, \beta) = \prod_{m=1}^{M} p(\vec{w_m}, \vec{z_m}|\alpha, \beta). \tag{1.8}$$

La probabilité de tous les mots du corpus est finalement déterminée par :

$$p(\vec{w}|\alpha, \beta) = \prod_{m=1}^{M} p(\vec{w_m}|\alpha, \beta). \tag{1.9}$$

On cherche à déterminer la probabilité a posteriori de $\vec{z}$, Φ, et Θ, sachant les mots $\vec{w}$ observés. Cependant, Φ et Θ peuvent être calculés directement à partir de l'association entre les $w_{m,n}$ observés et les $z_{m,n}$ correspondant, on les exclut donc de l'inférence. La probabilité

exacte que l'on recherche est donc la suivante :

$$p(\vec{z}|\vec{w},\alpha,\beta) = \frac{p(\vec{z},\vec{w}|\alpha,\beta)}{p(\vec{w}|,\alpha,\beta)} = \frac{\prod_{m=1}^{M} p(\vec{z_m},\vec{w_m}|\alpha,\beta)}{\prod_{m=1}^{M}\sum_{z_{m,n}} p(\vec{z_m},\vec{w_m}|\alpha,\beta)}. \tag{1.10}$$

Cette probabilité est incalculable de façon directe à cause de son dénominateur consistant en une somme sur K^M termes. Elle va donc être estimée à l'aide de l'échantillonnage de Gibbs, une technique issue des MCMC (Monte Carlo Markov Chain) (cf Annexe 3).

1.4.4 Échantillonnage de Gibbs

Cette technique est issue du constat qu'il est impossible d'inférer la distribution conjointe de toutes les variables du modèle simultanément. On va donc chercher, pour chaque variable à tour de rôle, à inférer sa distribution, conditionnellement à toutes les autres variables, puis à en tirer une nouvelle réalisation. En répétant cette opération sur toutes les variables un grand nombre de fois, la théorie montre que les réalisations tirées (c'est-à-dire l'échantillon) finissent par converger vers ce qui serait échantillonné à partir de la distribution conjointe cible. Ensuite, les propriétés de la distribution peuvent être calculées statistiquement à partir de l'échantillon.

L'échantillonnage de Gibbs se décompose en deux temps :
— Une étape d'initialisation où les mots sont assignés aléatoirement aux sujets
— Un processus itératif dans lequel, pour chaque mot $w_{m,n}$, on calcule la probabilité d'être affecté à chacun des sujets k, conditionnellement à l'affectation actuelle de tous les autres mots. On utilise ensuite ces probabilités pour réassigner les mots aux sujets à l'aide de la loi multinomiale. Cette étape est répétée jusqu'à ce que la condition d'arrêt choisie soit atteinte.

On peut ensuite calculer la matrice de distribution des termes par sujet Φ et la matrice de distribution des sujets par document Θ à partir des vecteurs $\vec{z}$ et $\vec{w}$.

Il est enfin possible d'affecter définitivement des mots aux sujets et des sujets aux textes à partir des matrices précédentes.

On modélise le fonctionnement de l'échantillonneur de Gibbs sur le schéma 1.9. Pour cela on définit :
— Le vecteur de vecteurs des affectations sujet-mot $\vec{z} = \{z_{ij}\}_{1\leq i\leq M, 1\leq j\leq n(i)}$, avec M le nombre de documents et n(i) le nombre de mots du document i
— Le vecteur de vecteurs des affectations sujet-mot exclu du sujet associé au mot $w_{i,j}$, $\vec{z_{-ij}} = \{z_{mp}\}_{1\leq m\leq M, 1\leq p\leq n(i), m\neq i, p\neq j}$
— Le vecteur de vecteurs représentant le corpus $\vec{w} = \{w_{ij}\}_{1\leq i\leq M, 1\leq j\leq n(i)}$
— Le vecteur des probabilités d'affectation du mot w_{ij} à chacun des K sujets $\vec{P_{z_{ij}}} = \left\{P^k_{z_{ij}}\right\}_{1\leq k\leq K}$
— N, représentant le nombre d'itérations de l'algorithme (un critère d'arrêt possible)

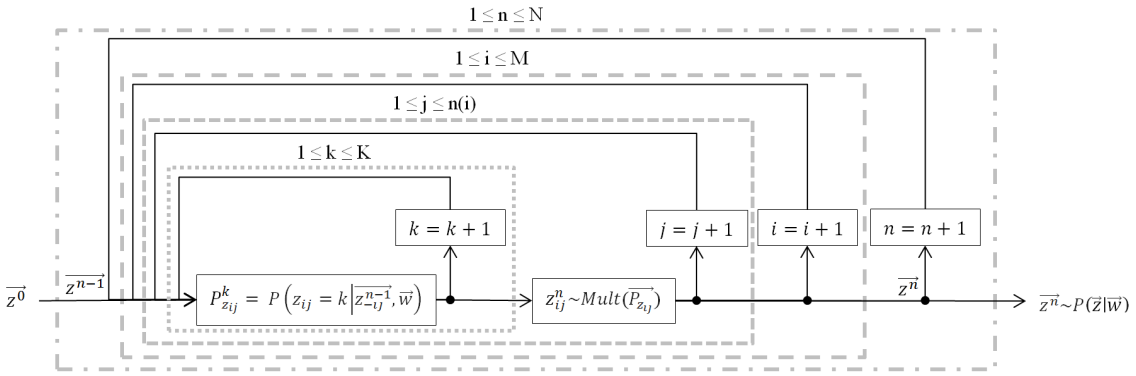


FIGURE 1.9 – Schéma de fonctionnement de l'échantillonneur de Gibbs appliqué au LDA

1.4.5 Perplexité

Un critère d'arrêt possible autre que le nombre d'itérations peut être basé sur la perplexité, sa valeur, ou son gradient d'une itération de l'échantillonneur à l'autre.

La perplexité [63] traduit la capacité d'un modèle à être généralisé à des données inconnues. Celle-ci est définie comme la réciproque de la moyenne géométrique par élément de la vraisemblance d'un corpus de test sachant le modèle avec ses données d'apprentissage actuelles. Plus concrètement, dans le cas du LDA, il s'agira de l'inverse de la vraisemblance moyenne par mot du corpus de test sachant l'état actuel des distributions Φ et Θ. La relation est la suivante :

$$perplexité(W_{test}) = \left[\prod_{m \in M} p(\vec{w_m}) \right]^{-\frac{1}{N}}. \tag{1.11}$$

W_{test} représente le corpus de test, $p(\vec{w_m})$ est la vraisemblance du document m, et N le nombre total de mots du corpus. L'indication qu'apporte la perplexité sur la qualité de l'apprentissage peut être interprétée comme suit : pour une position de mot considérée, la probabilité d'obtenir le mot réellement présent dans le corpus de test à l'aide du modèle sera l'inverse de la perplexité. Cette probabilité est à comparer à celle d'un tirage selon une loi uniforme, mais sur l'univers de tous les mots du dictionnaire. Par exemple, pour des mots de 8 bits, sans apprentissage, la probabilité d'obtenir le bon mot au hasard est de $1/256$. Pour un modèle initialisé sans a priori, la perplexité débutera à 256, et diminuera avec les itérations de l'échantillonneur.

1.4.6 Algorithme

L'Algorithme 4 présente sous forme de pseudo-code l'algorithme d'échantillonnage de Gibbs appliqué au LDA permettant de remonter aux variables latentes $\vec{z}$, Φ, et Θ, sachant le corpus W observé.

À noter que la notation -mn signifie tous les éléments de l'ensemble considéré, excepté celui en position mn.

Algorithme 4 : Algorithme LDA avec échantillonnage de Gibbs

Données : W,K,v,α,β
Résultat : $\vec{z}$, Φ, et Θ

Initialiser à 0 $n_m^{(k)}$, représentant le nombre de fois qu'un mot du document m est affecté au sujet k;
Initialiser à 0 n_m, représentant le nombre de mots du document m;
Initialiser à 0 $n_k^{(t)}$, représentant le nombre de fois que le terme t est affecté au sujet k;
Initialiser à 0 n_k, représentant le nombre de fois qu'un mot est affecté au sujet k;
pour *m = 1 à M* **faire**
    pour *n = 1 à* N_m **faire**
        $z_{m,n}$ = k $\sim$ multinomiale(1/K);
        $n_m^{(k)} + 1$, $n_m + 1$, $n_k^{(t)} + 1$, $n_k + 1$;
    fin
fin
tant que *Condition d'arrêt non atteinte* **faire**
    pour *m = 1 à M* **faire**
        pour *n = 1 à* N_m **faire**
            Le mot $w_{m,n}$ est un terme t assigné au sujet k;
            $n_m^{(k)} = n_m^{(k)}$ - 1;
            $n_m = n_m$ - 1;
            $n_k^{(t)} = n_k^{(t)}$ - 1;
            $n_k = n_k$ - 1;
            pour *i = 1 à K* **faire**
                On calcule chaque probabilité
                $$p(z_{mn} = i | z_{-\vec{mn}}, \vec{w}) \propto \frac{n_k^{(t)} + \beta}{n_k + V \times \beta} \times \frac{n_m^{(k)} + \alpha}{n_m + K \times \alpha};$$
            fin
            On normalise ces probabilités :
            $$p(z_{mn} = i | z_{-\vec{mn}}, \vec{w}) = \frac{p(z_{mn} = i | z_{-\vec{mn}}, \vec{w})}{\sum_{i=1}^{K} p(z_{mn} = i | z_{-\vec{mn}}, \vec{w})};$$
            On tire aléatoirement un sujet : k' $\sim$ multinomiale($p(z_{mn} | z_{-\vec{mn}}, \vec{w})$));
            Le mot $w_{m,n}$ est un terme t réassigné au sujet k';
            $n_m^{(k')} = n_m^{(k')} + 1$;
            $n_m = n_m + 1$;
            $n_{k'}^{(t)} = n_{k'}^{(t)} + 1$;
            $n_{k'} = n_{k'} + 1$;
        fin
    fin
fin
Calculer Φ, avec $\phi_{k,t} = \frac{n_k^{(t)} + \beta}{n_k + V \times \beta}$;
Calculer Θ, avec $\theta_{m,k} = \frac{n_m^{(k)} + \alpha}{n_m + K \times \alpha}$;

1.4.7 Application à la configuration d'étude de la thèse

L'algorithme utilisé dans la configuration d'étude de la thèse dérive du précédent, auquel nous avons apporté les modifications suivantes :

— Les documents considérés dans l'algorithme LDA sont remplacés par les messages DLL à analyser, le corpus devient donc la trace liaison de données. Les sujets sont remplacés par les mots-clés du protocole, et les mots par des n-grams, ensemble de n bits consécutifs. Les n-grams n'ayant pas de délimiteurs naturels comme dans le cas des mots, avec les espaces, on génère tous les n-grams possibles à partir de la trame à analyser. Les termes sont remplacés par les différentes séquences de n bits possibles. Cette équivalence est résumée dans la Table 1.3.

— Le dictionnaire est composé de tous les n-grams formables à partir de n bits.

— On utilise comme métrique d'arrêt le gradient de la perplexité. Cette dernière est calculée à partir de la logvraisemblance, afin d'éviter l'underflow, de la façon suivante :

$$perplexité(W_{apprentissage}) = exp - \frac{\sum\limits_{m \in M} logp(\vec{w_m})}{\sum\limits_{m \in M} N_m}, \tag{1.12}$$

avec M l'ensemble des trames de la trace liaison de données d'apprentissage, N_m le nombre total de n-grams dans la trame DLL m, et

$$p(\vec{w_m}) = \prod_{t \in V} (\sum_{k \in K} \phi_{k,t} \theta_{m,k})^{N_m^t}, \tag{1.13}$$

avec V l'ensemble des n-grams uniques, K l'ensemble des mots-clés, et N_m^t le nombre d'occurrences dans m du n-gram unique t. La formule de calcul du gradient de perplexité entre deux instants consécutifs n-1 et n est la suivante :

$$gradient\ de\ perplexité(n, n-1) = \frac{|perplexité(n) - perplexité(n-1)|}{perplexité(n)}. \tag{1.14}$$

On poursuit le rééchantillonnage tant que le gradient de la perplexité est au-dessus du gradient maximal de perplexité divisé par le nombre de trames de la trace. Le seuil utilisé dans la pratique est donc le suivant :

$$seuil = \frac{gradient\ maximal\ de\ perplexité}{nombre\ de\ trames\ de\ la\ trace}. \tag{1.15}$$

— Une fois les matrices de distribution des mots-clés et des n-grams obtenues, on sélectionne, pour chaque mot-clé, les n-grams dont les probabilités d'apparition sont les plus fortes. Pour cela, on classe les n-grams par probabilités décroissantes, puis on les parcourt en calculant le gradient entre chaque couple de probabilités consécutives. Un n-gram est sélectionné si le gradient entre lui et le n-gram précédent est inférieur au gradient maximal de probabilité. Le n-gram à la plus forte probabilité est automatiquement sélectionné. Si l'on considère un mot-clé k, et son vecteur de distribution de n-grams associé, ϕ_k, ordonné de façon décroissante, un n-gram en position n $\in [2, Card(\phi_k)]$ est donc sélectionné si et seulement si :

$$\frac{|\phi_k(n) - \phi_k(n-1)|}{\phi_k(n)} \leq gradient\ maximal\ de\ probabilité. \tag{1.16}$$

— Pour permettre la détection de champs de longueurs différentes, l'algorithme est exécuté plusieurs fois, pour un ensemble de valeurs de n, et toutes les séquences uniques extraites sur l'ensemble des itérations sont conservées pour analyse des performances.

Terme général employé dans l'exposition du principe	**Terme équivalent employé dans le cas de la configuration d'étude de la thèse**
Corpus	Trace liaison de données
Document	Trame
Sujet	Mot-clé
Mot	N-gram
Terme	Séquence de n bits

TABLE 1.3 – Récapitulatif des termes équivalents

Les paramètres utilisés pour la simulation seront les suivants :
— Longueur maximale des séquences recherchées (longueur des n-grams)
— Nombre de mots-clés recherchés
— Valeur du gradient maximal de perplexité avant arrêt
— Valeur du gradient maximal de probabilité de sélection des n-grams
— Paramètre de Dirichlet α
— Paramètre de Dirichlet β

1.5 Conclusion

Nous avons présenté dans ce chapitre les principes théoriques des trois algorithmes d'identification de séquence auxquels nous nous intéressons.

Le premier, Variance Distribution of the Variances (VDV), s'appuie sur une simple analyse à deux niveaux de la variance au sein de la trace à analyser : les parties de la trace présentant une variance des variances suffisamment faible sont sélectionnées, et leurs valeurs extraites en tant que séquences remarquables.

Le second, une version modifiée d'Aho-Corasick (AC), repose sur le calcul de la fréquence d'apparition des séquences au sein de la trace à l'aide d'une machine d'état. Les fréquences apparaissant suffisamment de fois sont sélectionnées, puis celles trop similaires sont fusionnées.

Enfin, le dernier, Latent Dirichlet Allocation (LDA), s'appuie sur un réseau Bayésien utilisé comme modèle génératif de la trace. Les paramètres du modèle sont appris via de l'échantillonnage de Gibbs, puis les séquences sont extraites à l'aide des distributions apprises.

Dans le prochain chapitre, nous allons comparer les performances de ces trois algorithmes, appliqués à des traces liaison de données.

Chapitre 2

Simulations comparatives des algorithmes VDV, AC et LDA

Nous allons maintenant comparer les performances des algorithmes VDV, AC et LDA, appliqués à des trames DLL d'un protocole très répandu de l'IoT : Zigbee [64]. Ce protocole est basé sur la norme IEEE 802.15.4 [65], extrêmement utilisée pour les couches physiques et liaison de données dans l'IoT.

Nous allons dans un premier temps présenter notre modèle de simulation, puis dans un second temps nos résultats.

2.1 Modèle de simulation

Nous allons tout d'abord présenter le modèle du simulateur avec lequel nous allons effectuer la simulation comparative des algorithmes.

2.1.1 Génération de la trace liaison de données

L'ordre dans lequel sont reçues les trames n'importe pas pour l'étape d'identification des séquences. Les trames seront donc aléatoirement générées à partir de la base de données des formats existants. À cette fin, ces trames seront modélisées par des objets constitués d'une suite de champs. Chacun de ceux-ci sera caractérisé par six paramètres :

— Sa trame mère
— Son champ parent
— Ses sous-champs
— Sa détectabilité
— Sa longueur
— Sa valeur

Par détectabilité, l'on entend la condition indiquant si la valeur du champ est censée être détectée par l'algorithme ou non. Le champ parent, les sous-champs, et la détectabilité sont inhérents au champ, et donc ne varient pas d'une instance à l'autre du champ. Il est à noter que si le champ est terminal, il ne possède pas de sous-champs. La longueur et la valeur peuvent être de deux types :

— Constant : identique pour chaque instance du champ.
— Variable : une valeur dans un ensemble fini est tirée aléatoirement, selon une loi normale discrétisée, pour chaque instance du champ.

Il est à noter que si le champ n'est pas terminal, il ne possède pas de valeur propre.

Tous les algorithmes simulés étant de type hors ligne (les trames ne sont pas analysées en temps réel), la génération de la trace liaison de données et l'exécution de chaque algorithme seront faites en différé.

À chaque génération de trame, l'un des formats de trame est tiré aléatoirement selon une loi uniforme discrète, puis cette trame génère son en-tête, sa charge utile, et son pied, qui à leur tour génèrent leurs propres sous-champs, et ainsi de suite, jusqu'aux champs terminaux qui génèrent les bits qu'ils contiennent. La Figure 2.1 ci-dessous présente ce principe de génération dans le cas particuliers de trois niveaux de champs. La trame est ensuite stockée avant d'être analysée par l'algorithme choisi.

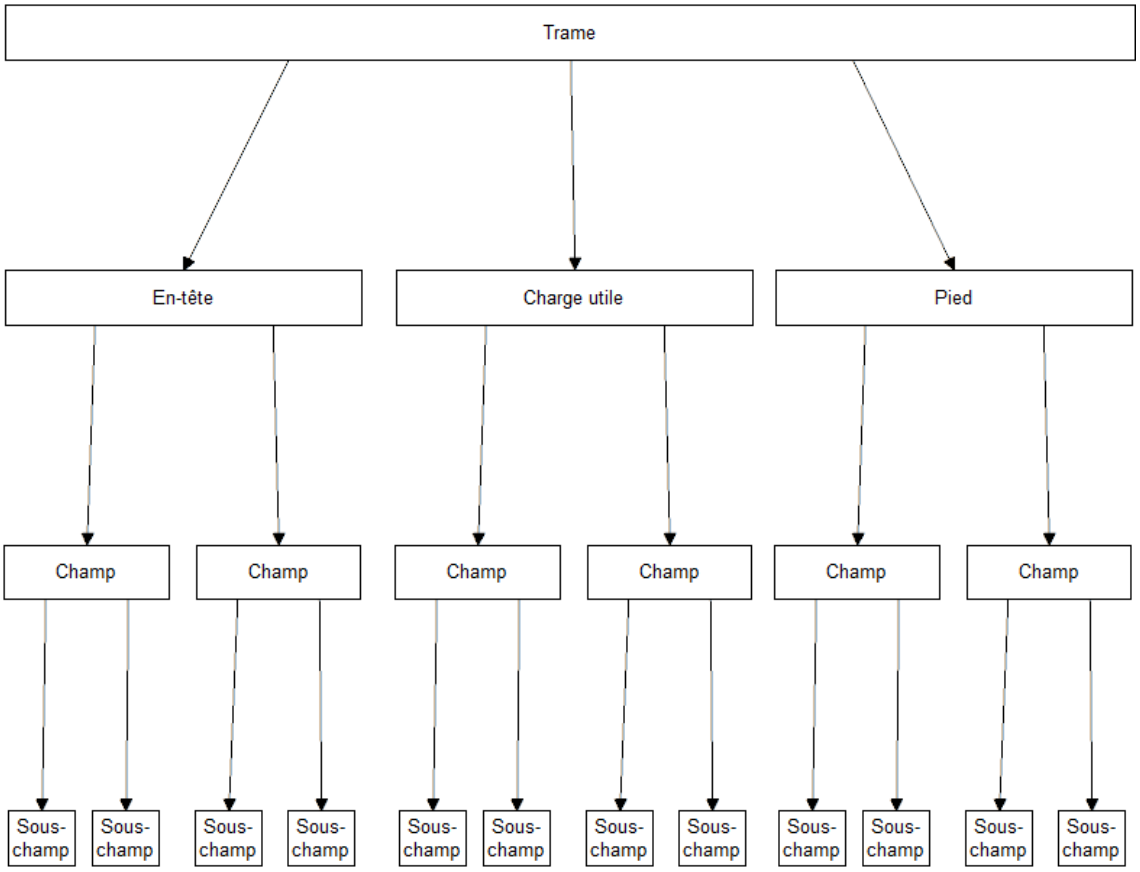


FIGURE 2.1 – Principe de génération des trames

2.1.2 Analyse de la trace liaison de données

Le schéma de la Figure 2.2 présente la procédure de calcul des performances des trois algorithmes à partir des trames générées comme présenté précédemment.

Les pré-traitements au VDV consistent à répartir aléatoirement les trames en flux selon une loi uniforme, et à effectuer le découpage de ces dernières en jetons. Les traitements complémentaires au VDV consistent à balayer toutes les trames analysées et à enregistrer en tant que séquence remarquable toutes les suites de bits différentes se trouvant aux sections trouvées par l'algorithme.

Les pré-traitements au LDA consistent à effectuer le découpage des trames en jetons. Les traitements complémentaires au LDA consistent à sélectionner au sein de chaque mot-clé les n-grams intéressants.

Les algorithmes (plus éventuellement un traitement complémentaire) fourniront en sortie les séquences de bits récurrentes et leurs positionnements possibles relativement au début de la trame. Ces données seront stockées dans un objet dédié.

2.1.3 Calcul des performances et comparaison

Afin de quantifier numériquement les performances des algorithmes étudiés, nous introduisons trois concepts :

- La **séquence** : Une séquence s est caractérisée par sa longueur l, sa valeur v, et les positions auxquelles elle apparaît p = $\{p_i\}_{i\in\mathbb{N}}$. Une séquence se représente donc par s(l,v,p). La liste des séquences S = $\{s\}$ est fournie par l'algorithme d'identification des séquences caractéristiques.
- Le **champ** : Un champ c est caractérisé par ses longueurs possibles L = $\{L_i\}_{i\in\mathbb{N}}$, ses valeurs caractéristiques V = $\{V_i\}_{i\in\mathbb{N}}$ (nul si la valeur est purement aléatoire), et les positions auxquelles il peut apparaître P = $\{P_i\}_{i\in\mathbb{N}}$. Un champ se représente donc par c(L,V,P). La liste des modèles de champs C = $\{c\}$ est générée à partir de la norme relative au protocole simulé (Zigbee, en l'occurence). Un champ peut être soit détectable ($V \neq \emptyset$), soit non détectable($V = \emptyset$).
- La **condition de coïncidence** : Les propriétés de s(l,v,p) et c(L,V,P) coïncident si la propriété suivante est vérifiée :

$$M(s,c) = \begin{cases} l \in L, v \in V, p \cap P \neq \emptyset, \ si\ V \neq \emptyset \\ l \in L, p \cap P \neq \emptyset, \ si\ V = \emptyset \end{cases}$$

Nous reprendrons les métriques de mesure des performances utilisées dans les articles étudiés [58, 57, 59], à savoir :

- La **précision**, quantifiant la part de ce qui a été bien détecté par rapport à tout ce qui a été détecté :

$$Précision = \frac{Vrais\ positifs}{Vrais\ positifs + Faux\ positifs}. \tag{2.1}$$

  Les vrais positifs sont les séquences correctement détectées, tandis que les faux positifs sont celles incorrectement détectées.

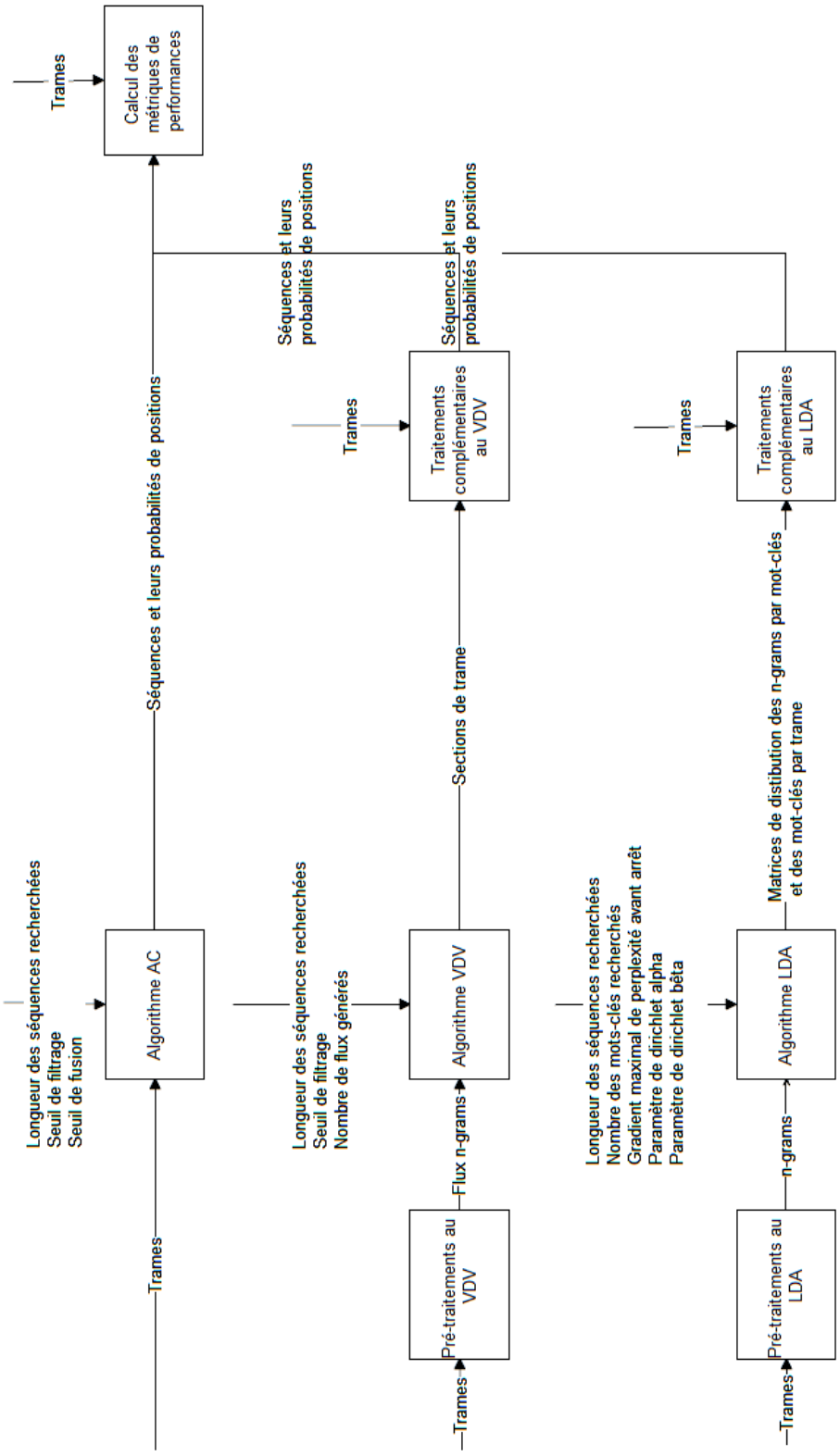


FIGURE 2.2 – Déroulement de l'analyse des trames

— La **couverture** (recall), quantifiant la part de ce qui a bien été détecté sur tout ce qu'il y avait à détecter :

$$Couverture = \frac{Vrais\ positifs}{Vrais\ positifs + Faux\ négatifs}. \tag{2.2}$$

Les faux négatifs sont les séquences incorrectement non-détectées.

Les séquences détectées sont classifiées de la façon suivante :
— Une séquence est un vrai positif si ses propriétés coïncident avec celles d'au moins un champ détectable.
— Une séquence est un faux positif dans tous les autres cas.
— Le nombre de faux négatifs est égal à la différence entre le nombre de séquences détectable tous champs confondus, et le nombre de vrais positifs.

Afin de pouvoir déterminer un compromis entre précision et couverture, nous emploierons le score F [66], qui est la moyenne harmonique de la précision et la couverture. Nous considérerons plus exactement le score F_1, pour lequel précision et couverture sont pondérées de façon égale.

$$F_1 = \frac{2 \times Précision \times Couverture}{Précision + Couverture} \tag{2.3}$$

Le score F_1 sera désigné par score F pour la suite de ce document.

Nous introduisons également une nouvelle notion, le **ratio de détection** des champs. Il s'agit de la quantification des informations d'un champ détectées.

On considère $S' = \{s \in S / \exists c \in C / M(s,c)\}$, l'ensemble des séquences coïncidant avec au moins un champ, et un champ générique $c_i(L_i, V_i, P_i)_{i\in\mathbb{I}}$, avec $L_i = \{L_{ij}\}_{j\in\mathbb{N}}, V_i = \{V_{ij}\}_{j\in\mathbb{N}}, P_i = \{P_{ij}\}_{j\in\mathbb{N}}$, avec $\mathbb{I}$ l'espace de tout les champs existant. On considère également une séquence générique $s_k(l_k, v_k, p_k)$.
Soit $q_{l_i} = Card(L'_i)/Card(L_i)$, le ratio du nombre de longueurs possibles de c_i détectées sur le nombre total de longueurs possibles de c_i, avec

$$L'_i = \left\{L_{ij} / \exists s_k \in S' / l_k = L_{ij}\right\}. \tag{2.4}$$

On calcule la moyenne du ratio de détection des longueurs sur tous les champs. Cette grandeur statistique constituera une de nos métriques. Elle est calculée selon la formule de l'expression 2.5 suivante :

$$\mu_{q_l} = \sum_{i\in I} \frac{q_{l_i}}{Card(I)}. \tag{2.5}$$

Soit $q_{v_i} = Card(V'_i)/Card(V_i)$, le ratio du nombre de valeurs possibles de c_i détectées sur le nombre total de valeurs possibles de c_i, avec

$$V'_i = \left\{V_{ij} / \exists s_k \in S' / v_k = V_{ij}\right\}. \tag{2.6}$$

On calcule la moyenne du ratio de détection des valeurs sur tous les champs. Cette grandeur statistique constituera une de nos métriques. Elle est calculée selon la formule de l'expression 2.7 suivante :

$$\mu_{q_v} = \sum_{i\in I} \frac{q_{v_i}}{Card(I)}. \tag{2.7}$$

Soit enfin $q_{p_i} = Card(P'_i)/Card(P_i)$, le ratio du nombre de positions possibles de c_i détectées sur le nombre total de positions possibles de c_i, avec

$$P'_i = \left\{ P_{ij} / \exists s_k \in S' / \exists p_k / P_{ij} \in p_k \cap P_i \right\}. \tag{2.8}$$

On calcule la moyenne du ratio de détection des positions sur tous les champs. Cette grandeur statistique constituera une de nos métriques. Elle est calculée selon la formule de l'expression 2.9 suivante :

$$\mu_{q_p} = \sum_{i \in I} \frac{q_{p_i}}{Card(I)}. \tag{2.9}$$

Les différentes métriques employées dans ce chapitre sont rassemblées de façon synthétique dans la Table 2.1.

Métriques employées	**Définition**
Précision	Proportion des séquences bien détectées par rapport à toutes les séquences détectées
Couverture	Proportion des séquences bien détectées sur toutes les séquences qu'il y avait à détecter
Score F	Moyenne harmonique de la précision et de la couverture
Ratio moyen de détection des longueurs	Proportion moyenne du nombre de longueurs possibles d'un champ détectées sur le nombre total de longueurs possibles du même champ
Ratio moyen de détection des valeurs	Proportion moyenne du nombre de valeurs possibles d'un champ détectées sur le nombre total de valeurs possibles du même champ
Ratio moyen de détection des positions	Proportion moyenne du nombre de positions possibles d'un champ détectées sur le nombre total de positions possibles du même champ

TABLE 2.1 – Récapitulatif des métriques employées

2.2 Déroulement des simulations

Nous étudierons dans cette simulation comparative l'influence des paramètres internes des algorithmes étudiés sur les performances de la détection des séquences en fonction du nombre de trames de la trace DLL générée.

Ces algorithmes se basant sur des analyses statistiques ou probabilistes, il nous a donc semblé indispensable de mener nos simulations sur une large gamme de tailles de trace liaison de données, afin d'en montrer l'impact. Cependant, pour des raisons de puissance de calcul de nos machines, nous avons décidé de nous limiter à 1000 trames par trace. Toutes nos campagnes de simulations emploierons donc un nombre de trames simulé variant de 1 à 1000 par pas logarithmique, afin de couvrir le domaine avec peu de points.

Nous savons que la majorité des séquences à trouver dans notre trace liaison de données sont inférieures ou égales à 8. Nous avons également constaté une forte augmentation de la puissance de calcul nécessaire lorsque nous cherchons des séquences de 16 bits, les plus longues

présentes dans la trace. Nous avons donc décidé de fixer la longueur maximale des séquences à détecter à 8, et lors de l'étude de l'influence de ce paramètre, nous le ferons varier de 1 à 8 par pas en puissance de deux.

La simulation comparative se déroulera en deux temps :
— Observation de l'influence de chacun des paramètres des algorithmes en les simulant sur un domaine arbitraire mais judicieusement choisi de jeux de paramètres.
— Choix du jeu de paramètres le plus performant parmi tous ceux qui ont été simulés, et comparer les performances obtenues.

Cette méthode a l'avantage de ne pas nécessiter un grand nombre de simulations et de permettre d'obtenir des jeux de paramètres donnant de bonnes performances, mais en contrepartie, il n'est pas possible de trouver les meilleures performances que puissent atteindre les algorithmes. Ceci est dû au fait que l'approche d'optimisation employée est simple, car on considère que tous les paramètres influencent les algorithmes de manière indépendante.

Chaque simulation pour un jeu de paramètres donné sera effectuée 100 fois, de manière à lisser les résultats. Les métriques utilisées seront celles rassemblées dans la Table 2.1.

2.3 Simulations de l'algorithme VDV

Nous allons dans un premier temps présenter les valeurs des paramètres de l'algorithme VDV employées dans chacune de nos campagnes de simulations, puis discuter des résultats obtenus.

2.3.1 Paramètres

La Table 2.2 ci-dessous rassemble toutes les différentes combinaisons de valeurs de paramètres pour lesquelles chaque campagne de simulations a été effectuée. Chaque colonne représente une campagne de simulations visant à étudier l'influence d'un paramètre sur les performances de l'algorithme.

Paramètres	**Campagne 1**	**Campagne 2**	**Campagne 3**
Nombre de trames	1 à 1000 par pas logarithmique	1 à 1000 par pas logarithmique	1 à 1000 par pas logarithmique
Longueur des séquences	1 à 8 par pas en puissance de deux	8	8
Nombre de flux	10	5 à 30 par pas de 5	10
Seuil de filtrage du VDV	1	1	0.0001 à 1 par pas en puissance de dix

TABLE 2.2 – Valeurs des paramètres des simulations de VDV effectuées

2.3.2 Influence de la longueur maximale des séquences recherchées

Sur le graphique de la Figure 2.3, on constate tout d'abord que la précision de l'algorithme VDV décroit lorsque la longueur maximale des séquences recherchées augmente. Cela s'ex-

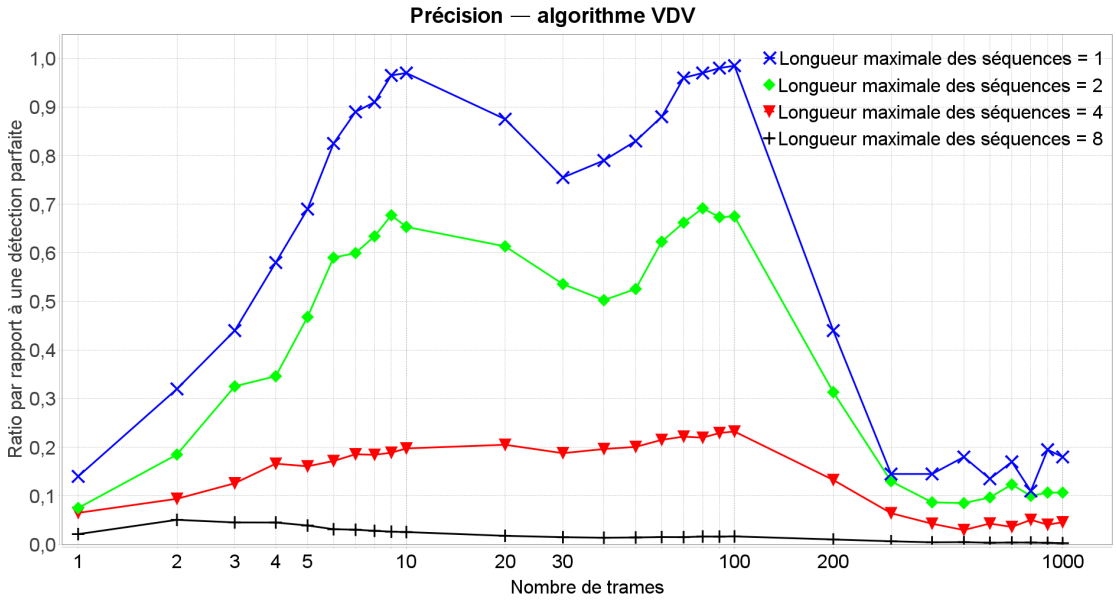


FIGURE 2.3 – Précision de l'algorithme VDV en fonction du nombre de trames et paramétré par la longueur maximale des séquences

plique par le fait que plus nous recherchons des séquences longues, plus les séquences possibles détectables sont nombreuses, et donc plus leurs nombres d'apparitions respectifs sont faibles. Par conséquent, le seuillage statistique est moins performant, d'où une augmentation des faux positifs.

On note également une rapide augmentation de la précision avec le nombre de trames dans la trace liaison de données, lorsque celui-ci est faible (en dessous de 10 trames). Ce comportement était prévisible, car il s'agit d'un filtrage statistique, donc il n'a de véritable sens que sur des échantillons de taille importante. Cependant, l'on constate aussi une chute de la précision lorsque la taille de la trace devient importante (au-delà de 100 trames). Il est dû au fait que lorsque la taille de la trace augmente, la variance diminue ; il devient donc plus difficile de discriminer les séquences à détecter, d'où une augmentation des faux positifs et diminution des vrais positifs.

On constate sur le graphique de la Figure 2.4 que la couverture de l'algorithme VDV croit avec l'augmentation de la longueur maximale des séquences recherchées. Cela paraît logique, puisque qu'avec l'augmentation de celle-ci, il devient possible de détecter plus de séquences remarquables présentes dans la trace liaison de données.

On remarque également un comportement identique à celui de la précision relativement au nombre de trames de la trace, ce qui s'explique par les mêmes raisons.

À l'aide du graphique de la Figure 2.5, nous pouvons conclure que pour 10 flux et un seuil de filtrage de 1, la longueur maximale des séquences recherchées maximisant les performances de l'algorithme VDV est de 4 bits, car nous favorisons les plus grandes traces DLL.

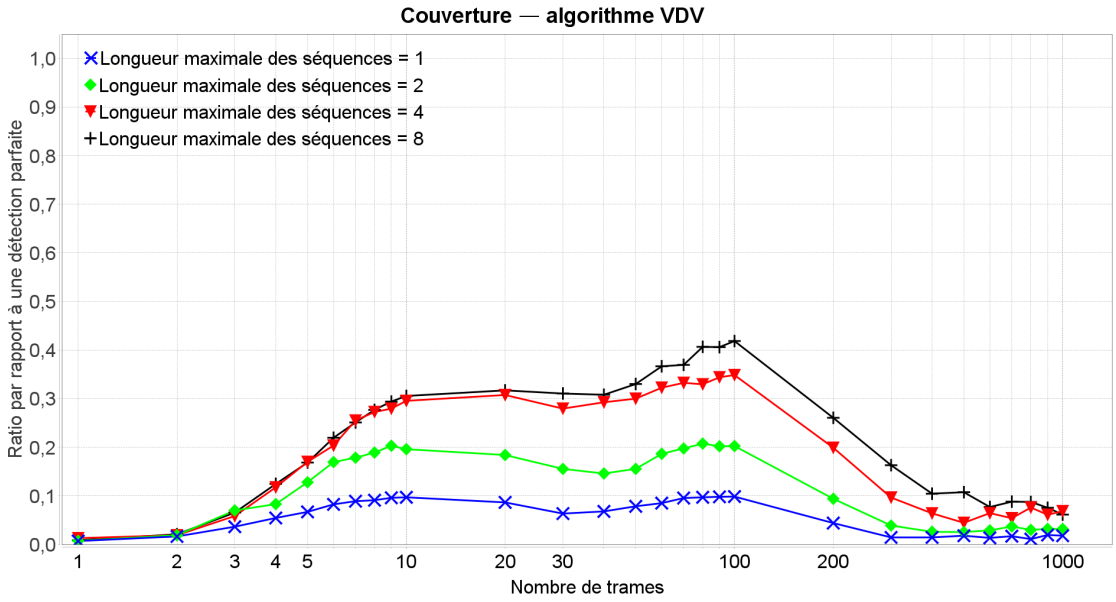


FIGURE 2.4 – Couverture de l'algorithme VDV en fonction du nombre de trames et paramétré par la longueur maximale des séquences

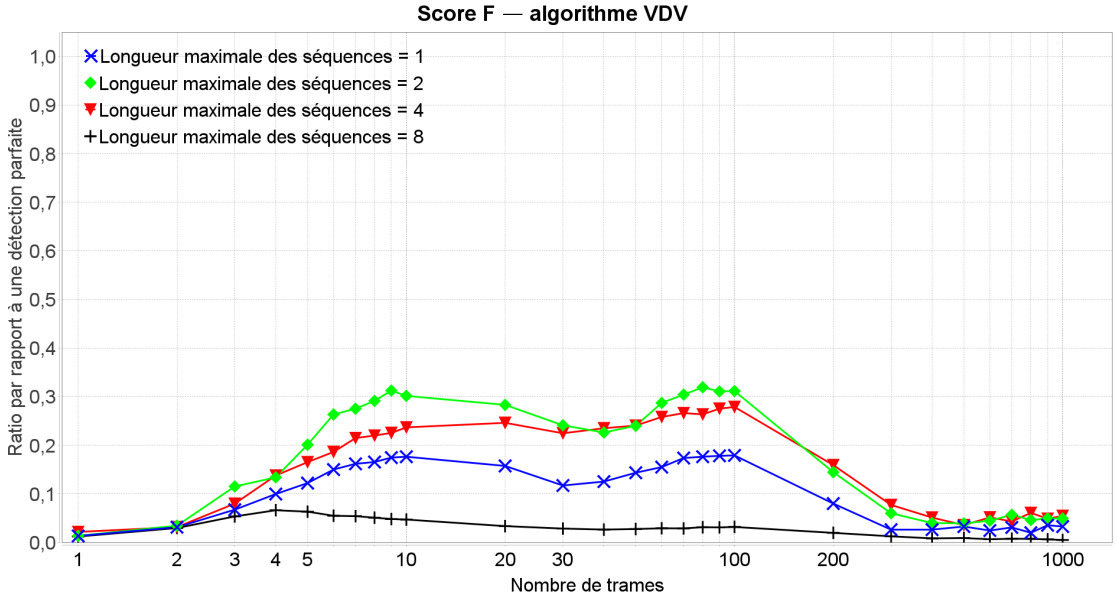


FIGURE 2.5 – Score F de l'algorithme VDV en fonction du nombre de trames et paramétré par la longueur maximale des séquences

2.3.3 Influence du nombre de flux générés

Les courbes de précision de l'algorithme VDV paramétrées par le nombre de flux générés en son sein ne seront pas présentées car ce paramètre semble n'avoir aucune influence sur cette métrique.

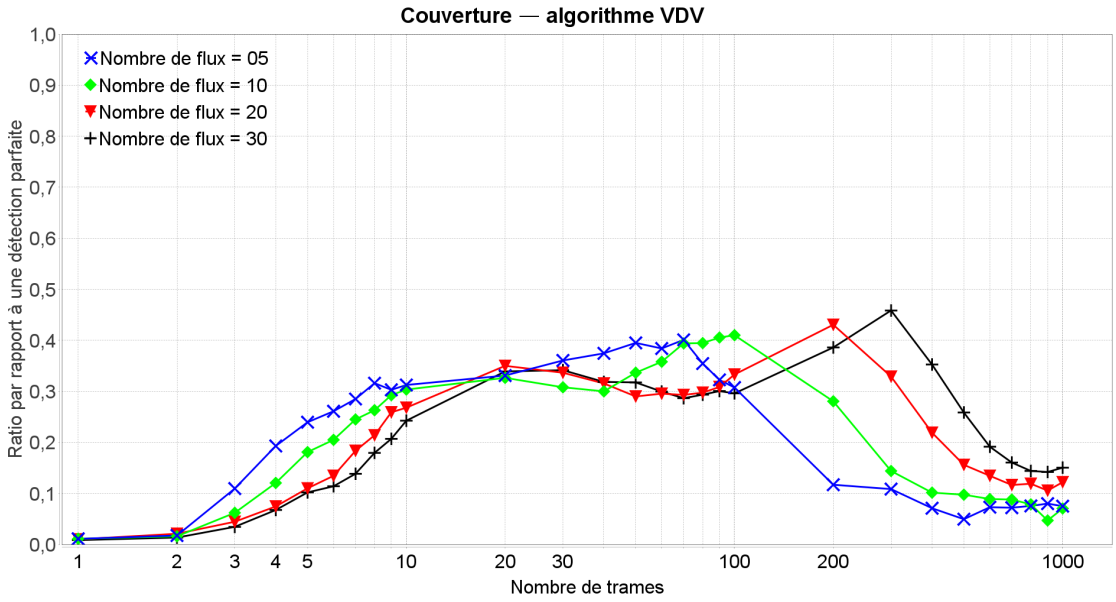


FIGURE 2.6 – Couverture de l'algorithme VDV en fonction du nombre de trames et paramétré par son nombre de flux

On peut scinder le graphique de la Figure 2.6 en trois parties selon l'axe du nombre de trames de la trace liaison de données. En dessous de 20 trames, la couverture de l'algorithme VDV décroit avec l'augmentation du nombre de trames. Cela s'explique par le fait que les calculs de variance effectués au sein de chaque flux perdent de leur sens si les flux contiennent trop peu de trames, d'où une meilleure couverture pour un plus faible nombre de flux.

De 20 à 100 trames, les performances sont à peu près équivalentes, car il s'agit d'une zone intermédiaire entre petites et grandes traces.

De 100 à 1000 trames, la couverture croit avec le nombre de flux, car en augmentant la diversité des flux, on tire pleinement parti du calcul de variance à deux niveaux effectué.

On remarque également un comportement identique à celui des courbes paramétrées par la longueur maximale des séquences recherchées (Figures 2.3 et 2.4), ce qui s'explique par les mêmes raisons.

Les courbes de score F étant quasi identiques à celles de couverture, nous pouvons donc dire que pour une longueur maximale des séquences recherchées de 8 et un seuil de filtrage de 1, le meilleur nombre de flux est de 30 ou plus, car on favorise les plus grandes traces DLL.

2.3.4 Influence du seuil de filtrage des séquences

On constate sur le graphique de la Figure 2.7 que la précision de l'algorithme VDV décroit avec l'augmentation du seuil de filtrage de l'algorithme VDV. Cela est dû au fait que lorsque le seuil de filtrage diminue, seules les séquences présentant la plus faible variance des variances sont retenues, et il s'agit généralement d'une séquence à détecter.

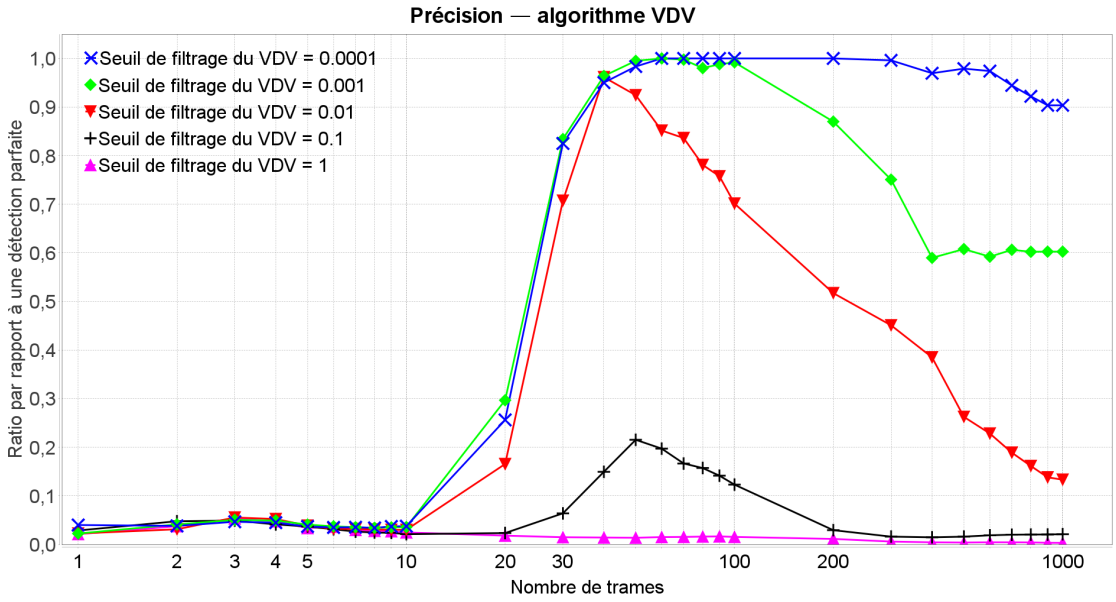


FIGURE 2.7 – Précision de l'algorithme VDV en fonction du nombre de trames et paramétré par son seuil de filtrage

On observe le même comportement relativement au nombre de trames que pour toutes les autres courbes de performances de l'algorithme VDV, mais on voit que la valeur du seuil de filtrage est bien le paramètre influant sur la taille limite de la trace liaison de données à partir de laquelle les performances chutent. Il aurait probablement été plus judicieux de lier ce seuil au nombre de trames présentes dans la trace, mais l'algorithme VDV utilisé comme référence ne présentant pas une telle fonctionnalité, nous ne l'avons pas implémentée.

La précision identique quel que soit le seuil de filtrage pour une trace de moins de 10 trames est lié au fait que la variance est tellement grande qu'une importante partie des jetons sont détectés, générant donc énormément de faux positifs.

On retrouve sur le graphique de la Figure 2.8 la chute des performances lorsque le seuil de filtrage est trop haut, caractéristique de notre implémentation de l'algorithme VDV. Cependant, on peut aussi confirmer qu'un seuil non dépendant de la taille de la trace n'est pas du tout pertinent ; en effet, avant 8 trames, toutes les courbes affichent la même croissance rapide, mais ensuite, on peut voir que chacune d'entre elles semble avoir approximativement le même comportement : une phase de décroissance, puis de croissance, et enfin de décroissance, mais décalée sur l'axe du nombre de trames de la trace. On peut ainsi donc supposer qu'à chaque taille de trace correspond un seuil de filtrage optimal.

À l'aide du graphique de la Figure 2.9, on peut conclure que pour une longueur maximale des séquences recherchées de 8 et 10 flux, le seuil de filtrage maximisant les performances de l'algorithme VDV sur l'intervalle simulé est de 0.001, car l'on privilégie les plus grandes traces. Cependant, si l'on devait considérer une trace plus importante, un seuil de filtrage de 0.0001 serait probablement une meilleure alternative, même si l'idéal serait d'indexer le seuil

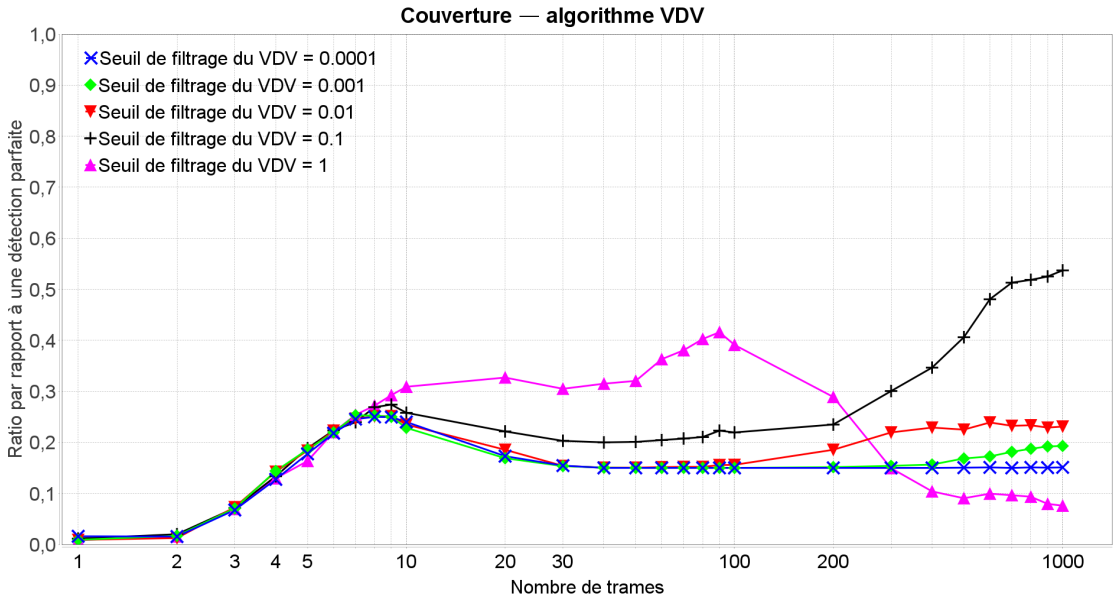


FIGURE 2.8 – Couverture de l'algorithme VDV en fonction du nombre de trames et paramétré par son seuil de filtrage

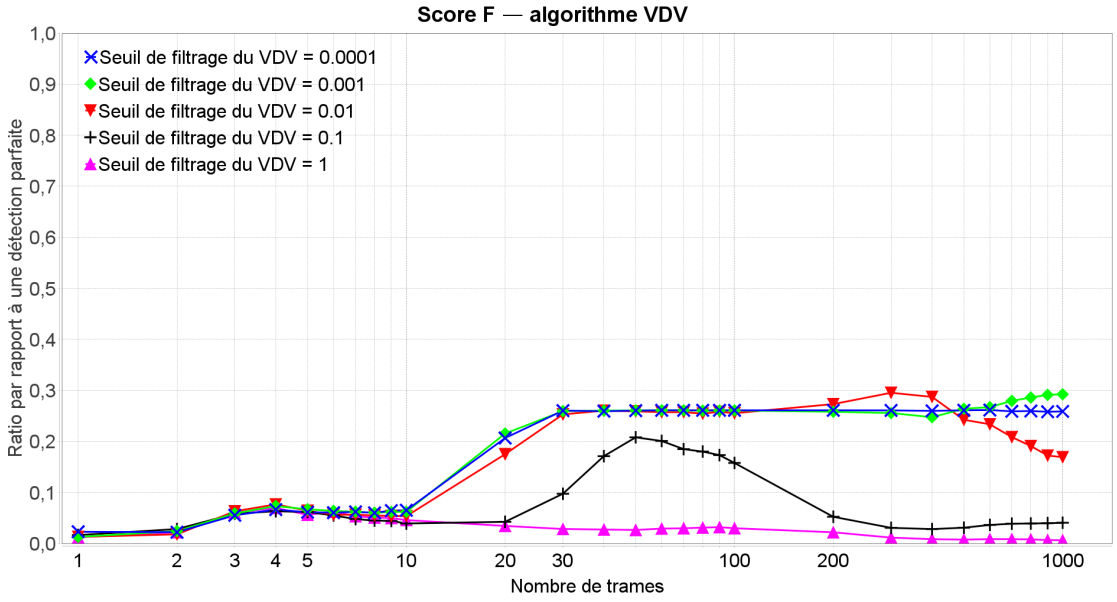


FIGURE 2.9 – Score F de l'algorithme VDV en fonction du nombre de trames et paramétré par son seuil de filtrage

de filtrage sur la taille de la trace liaison de données.

2.4 Simulations de l'algorithme AC

Nous allons maintenant présenter les valeurs des paramètres de l'algorithme AC employées dans chacune de nos campagnes de simulations, puis discuter des résultats obtenus.

2.4.1 Paramètres

La Table 2.3 ci-dessous rassemble toutes les différentes combinaisons de valeurs de paramètres pour lesquelles chaque campagne de simulations a été effectuée. Chaque colonne représente une campagne de simulations visant à étudier l'influence d'un paramètre sur les performances de l'algorithme.

Paramètres	**Campagne 1**	**Campagne 2**	**Campagne 3**
Nombre de trames	1 à 1000 par pas logarithmique	1 à 1000 par pas logarithmique	1 à 1000 par pas logarithmique
Longueur des séquences	1 à 8 par pas en puissance de deux	8	8
Seuil de filtrage de l'AC	1	0.7 à 1.5 par pas de 0.1	1
Seuil de fusion	1	1	0.1 à 1 par pas de 0.1

TABLE 2.3 – Valeurs des paramètres des simulations d'AC effectuées

2.4.2 Influence de la longueur maximale des séquences recherchées

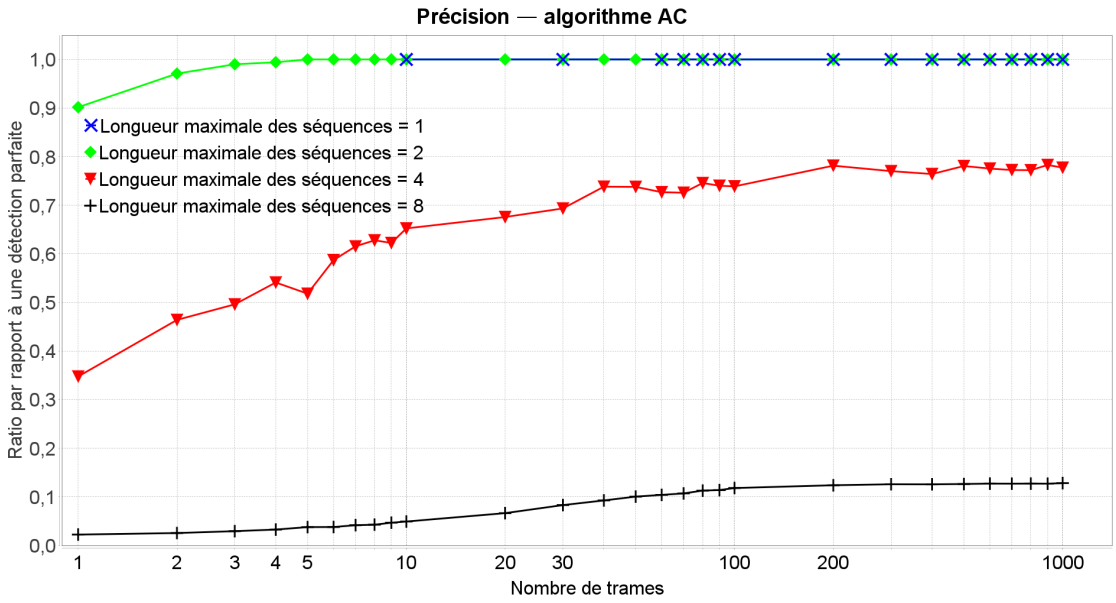


FIGURE 2.10 – Précision de l'algorithme AC en fonction du nombre de trames et paramétré par la longueur des séquences

On observe sur le graphique de la Figure 2.10 que la précision de l'algorithme AC décroit lorsque la longueur maximale des séquences recherchées augmente, comme pour l'algorithme VDV, et pour les mêmes raisons.

L'absence de points pour les petites tailles de trace de la courbe correspondant à une longueur maximale de séquence de 1 bit signifie qu'aucune séquence n'a été détectée pour au moins une des 100 répétitions de la simulation. Ce n'est pas surprenant, car l'application de statistiques à de très courtes séquences sur un petit nombre de trames ne tend pas à faire apparaître de tendances, de sorte que le filtrage peut aboutir à ce que rien ne soit sélectionné.

On voit également que la précision croit de façon logarithmique avec la taille de la trace liaison de données, mais, contrairement au VDV, ses performances ne finissent pas par chuter lorsque le nombre de trames augmente. Cela s'explique en comparant les formules de calcul des seuils de filtrage de ces deux algorithmes : celui de VDV ne dépend pas de la taille de la trace, alors que celui d'AC en dépend. Comme nous l'avions supposé dans la partie précédente, un tel seuil est plus pertinent.

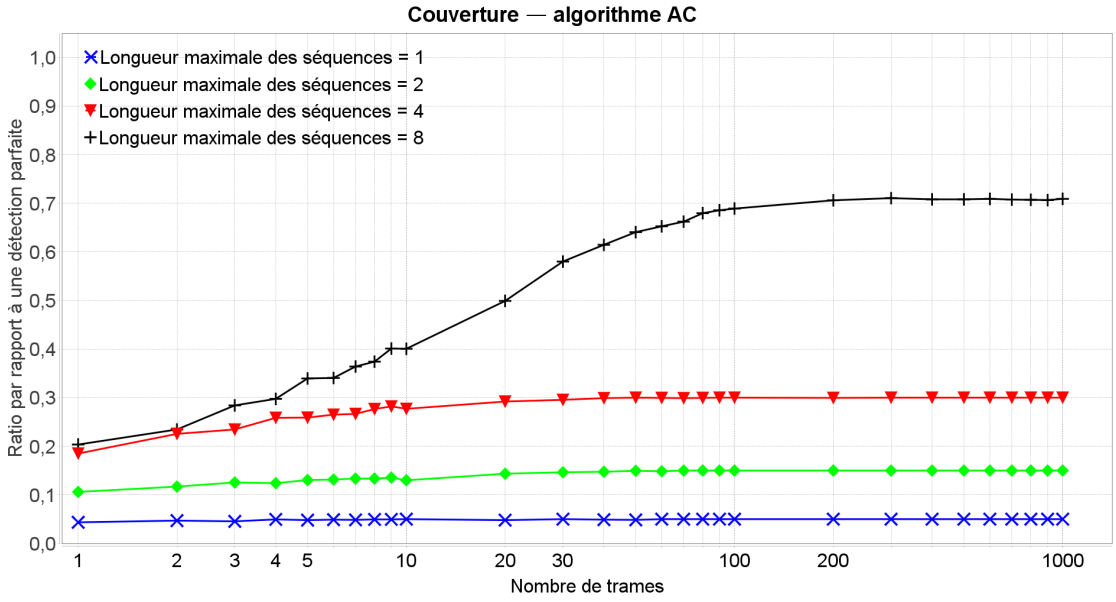


FIGURE 2.11 – Couverture de l'algorithme AC en fonction du nombre de trames et paramétré par la longueur des séquences

Sur le graphique de la Figure 2.11, on constate que la couverture de l'algorithme AC croit avec l'augmentation de la longueur maximale des séquences recherchées, comme pour l'algorithme VDV, et pour les mêmes raisons.

On observe également la même croissance logarithmique que pour les courbes précédentes sur la Figure 2.10, et pour les mêmes raisons.

À l'aide du graphique de la Figure 2.12, nous pouvons conclure que pour un seuil de filtrage et de fusion de 1, la longueur maximale des séquences recherchées maximisant les performances

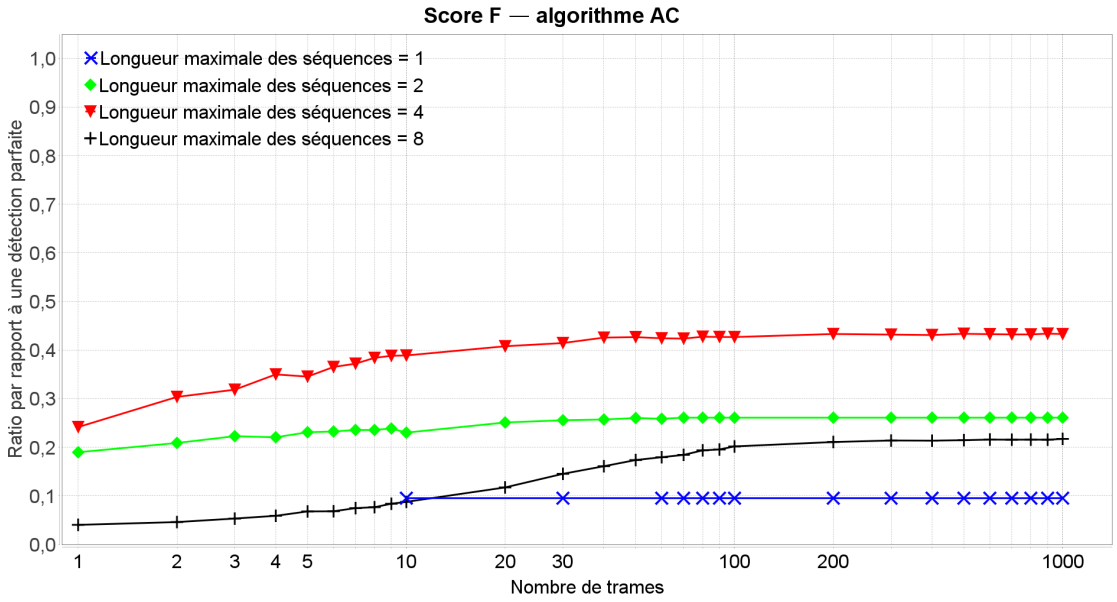


FIGURE 2.12 – Score F de l'algorithme AC en fonction du nombre de trames et paramétré par la longueur des séquences

de l'algorithme AC est de 4 bits.

2.4.3 Influence du seuil de filtrage des séquences

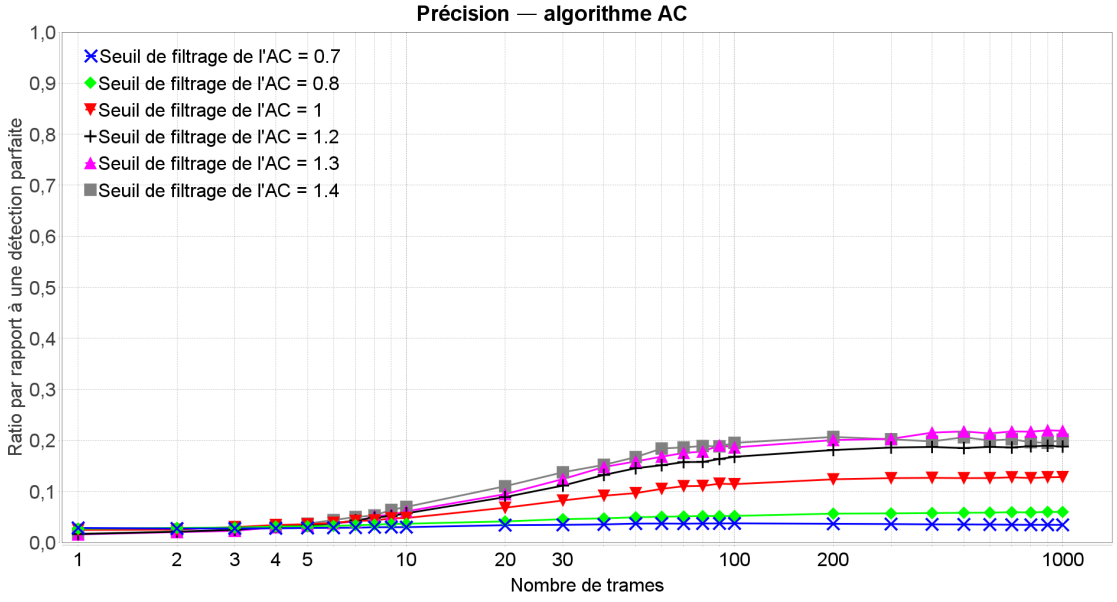


FIGURE 2.13 – Précision de l'algorithme AC en fonction du nombre de trames et paramétré par son seuil de filtrage

Le graphique de la Figure 2.13 nous montre que la précision de l'algorithme AC croit avec

le seuil de filtrage dans un premier temps, puis, au-delà de 1.3, décroit. L'augmentation du seuil de filtrage signifie que le filtrage est plus restrictif. Par conséquent, dans un premier temps, les séquences précédemment retenues que l'on ne retient désormais plus sont principalement des faux positifs, d'où une amélioration de la précision. Cependant, passé un certain niveau, le filtrage devient trop dur, et supprime plus de vrais positifs que de faux positifs, ce qui explique la présence de cet extremum.

On observe également la même croissance logarithmique que pour toutes les courbes de performances d'AC, et pour les mêmes raisons.

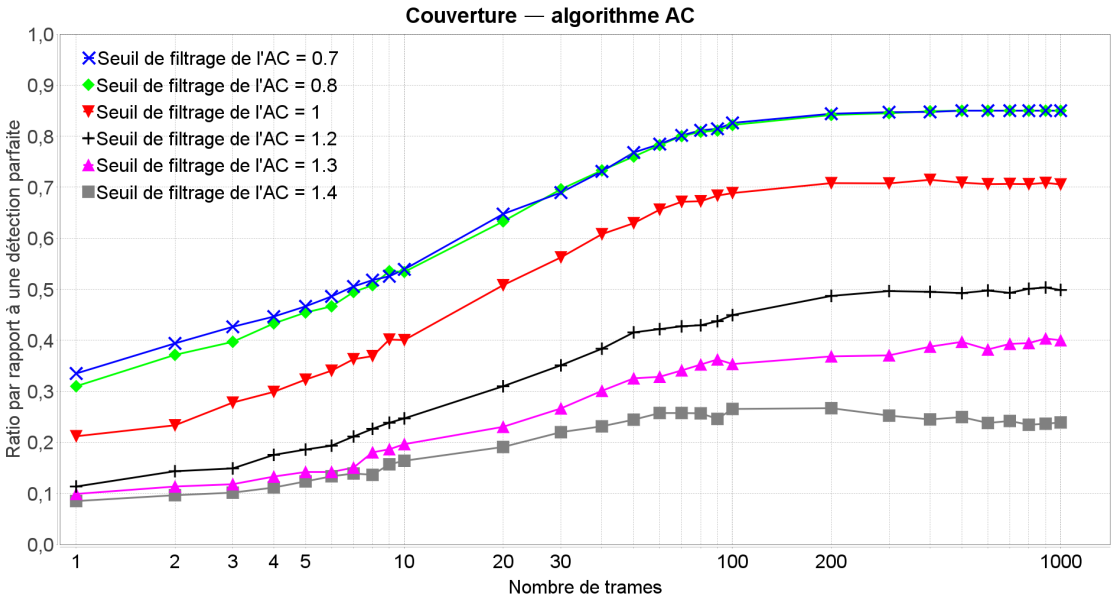


FIGURE 2.14 – Couverture de l'algorithme AC en fonction du nombre de trames et paramétré par son seuil de filtrage

On peut voir sur le graphique de la Figure 2.14 que la couverture de l'algorithme AC décroit lorsque la valeur du seuil de filtrage augmente, mais seulement à partir d'une valeur de 0.8, car la couverture cesse de s'améliorer en-deçà. Cela signifie simplement que toutes les séquences d'une longueur inférieure ou égale à 8 bits à détecter ont été détectées. Les 15% manquants sont causés par les trois séquences de 16 bits présentes dans la trace liaison de données.

À l'aide du graphique de la Figure 2.15, nous pouvons conclure que pour une longueur maximale des séquences recherchées de 8 et un seuil de fusion de 1, la valeur du seuil de filtrage maximisant les performances de l'algorithme AC est de 1,2 ou 1,3.

2.4.4 Influence du seuil de fusion des séquences

Le graphique de la Figure 2.16 nous montre que la précision de l'algorithme AC décroit avec l'augmentation du seuil de fusion entre 0.2 et 0.9, et est stable en dehors de ces bornes. Un faible seuil de fusion signifie que des séquences relativement différentes sont fusionnées, donc qu'il y a diminution du nombre total de séquences détectées. Il semble, d'après le gra-

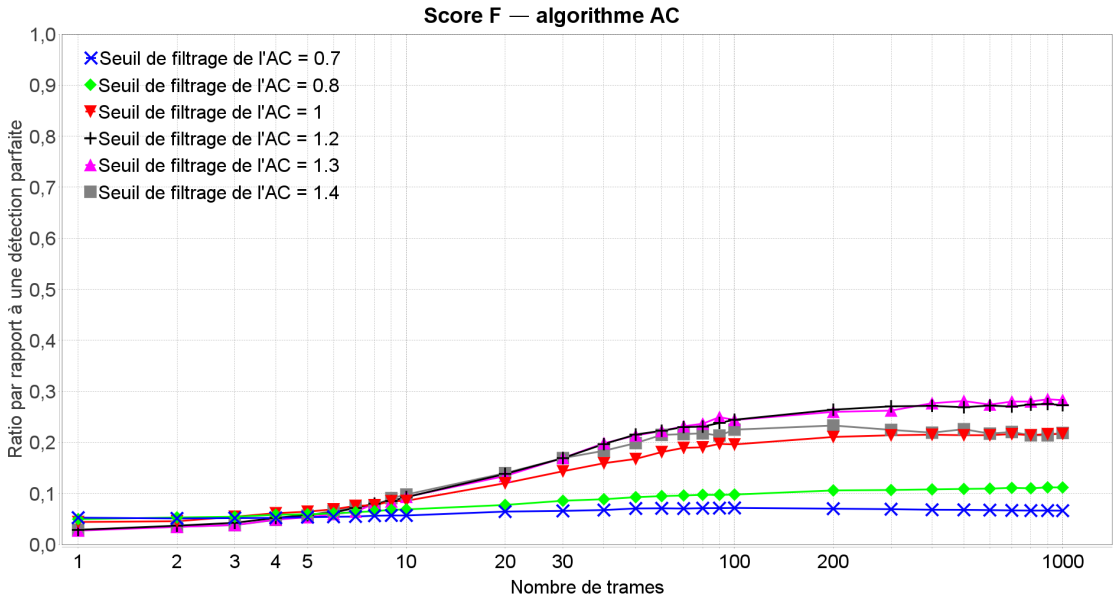


FIGURE 2.15 – Couverture de l'algorithme AC en fonction du nombre de trames et paramétré par son seuil de filtrage

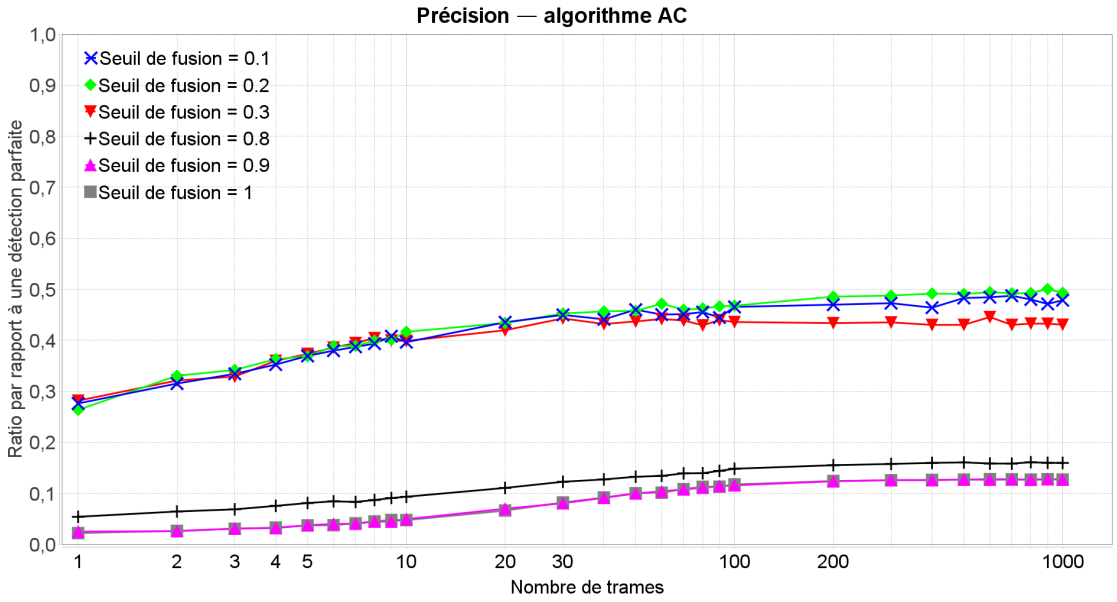


FIGURE 2.16 – Précision de l'algorithme AC en fonction du nombre de trames et paramétré par son seuil de fusion

phique, que le nombre de faux positif diminue plus rapidement que celui de vrais positifs, donc que l'on assimile les faux positifs aux vrais positifs, causant une augmentation de la précision. Cependant, lorsque le seuil atteint 0.2, pour chaque longueur, toutes les séquences sont fusionnées en une seule, d'où une impossibilité de gagner quoi que ce soit de plus par fusion. Au contraire, lorsque le seuil atteint 0.9, plus aucune séquence n'est fusionnable avec

une autre, d'où un impact inexistant de la fusion.

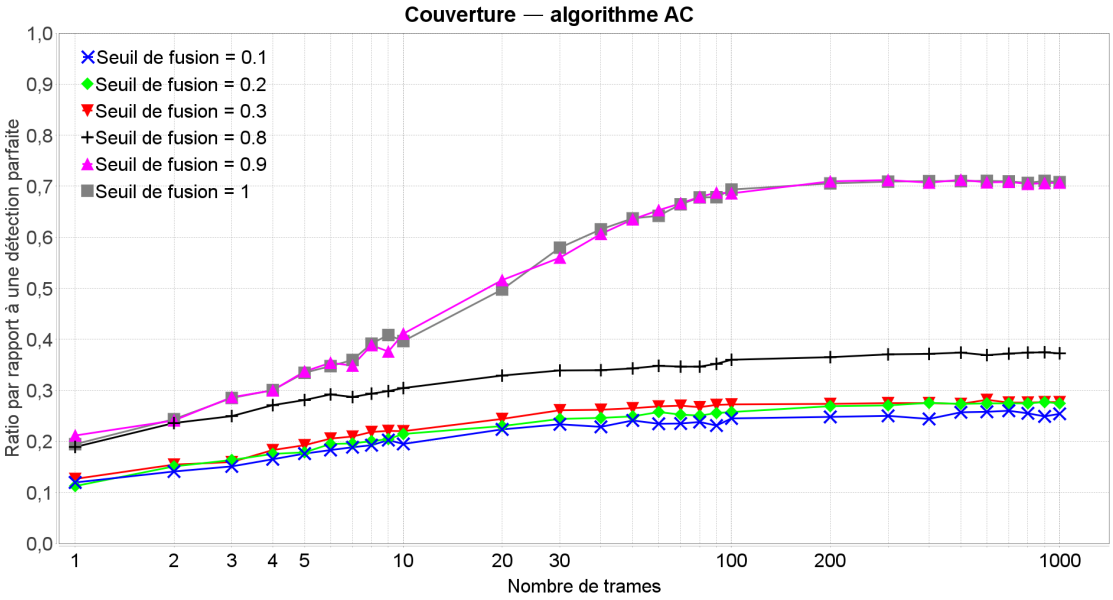


FIGURE 2.17 – Couverture de l'algorithme AC en fonction du nombre de trames et paramétré par son seuil de fusion

On constate sur le graphique de la Figure 2.17 que la couverture de l'algorithme AC augmente avec le seuil de fusion entre 0.2 et 0.9, et est stable en dehors de ces bornes. Cela paraît logique, car un plus haut seuil de fusion implique moins de fusion de séquences, donc une plus grande diversité, induisant une meilleure couverture. Les bornes visibles sont dues aux mêmes raisons que pour le graphique précédent de la Figure 2.16.

À l'aide du graphique de la Figure 2.18, nous pouvons conclure que pour une longueur maximale de 8 des séquences recherchées et un seuil de filtrage de 1, le seuil de fusion des séquences maximisant la performance de l'algorithme AC est de 0,2.

2.5 Simulations de l'algorithme LDA

Nous allons dans un premier temps présenter les valeurs des paramètres de l'algorithme LDA employées dans chacune de nos campagnes de simulations, puis discuter des résultats obtenus.

2.5.1 Paramètres

La Table 2.4 ci-dessous rassemble toutes les différentes combinaisons de valeurs de paramètres pour lesquelles chaque campagne de simulations a été effectuée. Chaque colonne représente une campagne de simulations visant à étudier l'influence d'un paramètre sur les performances de l'algorithme.

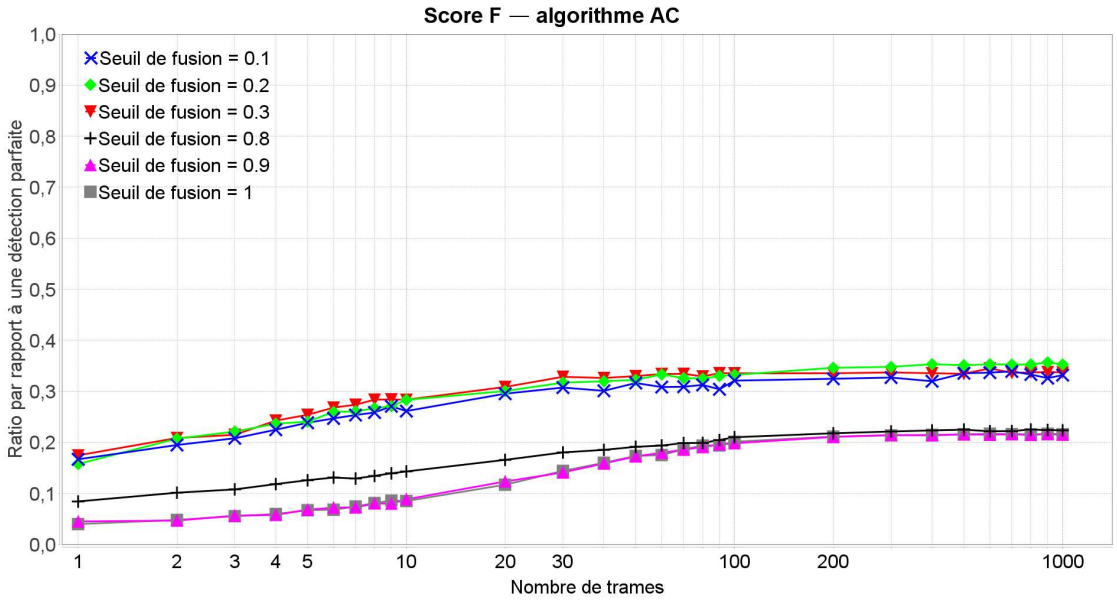


FIGURE 2.18 – Score F de l'algorithme AC en fonction du nombre de trames et paramétré par son seuil de fusion

2.5.2 Influence de la longueur maximale des séquences recherchées

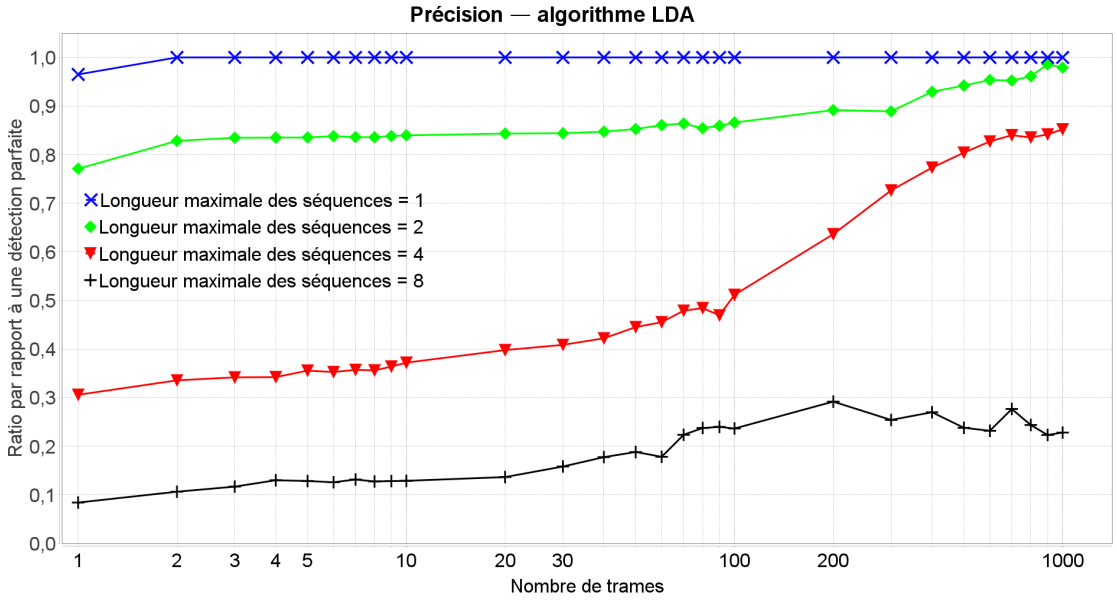


FIGURE 2.19 – Précision de l'algorithme LDA en fonction du nombre de trames et paramétré par la longueur maximale des séquences

On observe sur le graphique de la Figure 2.19 que la précision de l'algorithme LDA décroit lorsque la longueur maximale des séquences recherchées augmente, comme pour les algorithmes VDV et AC, et pour les mêmes raisons.

Paramètres	**Campagne 1**	**Campagne 2**	**Campagne 3**	**Campagne 4**	**Campagne 5**	**Campagne 6**
Nombre de trames	1 à 1000 par pas logarithmique	1 à 1000 par pas logarithmique	1 à 1000 par pas logarithmique	1 à 1000 par pas logarithmique	1 à 1000 par pas logarithmique	1 à 1000 par pas logarithmique
Longueur des séquences	1 à 8 par pas en puissance de deux	8	8	8	8	8
Nombre de mots-clés	10	3 5 10 20 30	10	10	10	10
Gradient maximal de perplexité	1	1	0.1 à 10 par pas logarithmique	1	1	1
Gradient maximal de probabilité	0.1	0.1	0.1	0.01 à 1 par pas logarithmique	0.1	0.1
Alpha	1	1	1	1	0.01 à 100 par pas en puissance de dix	1
Bêta	0.0001	0.0001	0.0001	0.0001	0.0001	0.000001 à 0.01 par pas en puissance de dix

TABLE 2.4 – Valeur des paramètres des simulations de LDA effectuées

On constate également une croissance approximativement linéaire, avec l'augmentation de la taille de la trace liaison de données, ce qui indique que, contrairement à l'algorithme AC, une technique basée sur l'apprentissage continue d'accroitre ses performances lorsque sa base d'apprentissage gagne en ampleur. Elle est donc plus adaptée pour un large volume de données.

Sur le graphique de la Figure 2.20, on constate que la couverture de l'algorithme LDA croit lorsque la longueur maximale des séquences recherchées augmente, comme pour les algorithmes VDV et AC, et ceci pour les mêmes raisons.

On remarque une phase de légère croissance suivie d'une légère décroissance de la couverture, lorsque la taille de la trace liaison de données augmente. Cela peut s'expliquer par la recherche d'un nombre de mots-clés trop faible, ce qui provoque la convergence de la plupart de ceux-ci vers un nombre limité de séquences, lorsque le nombre de trames augmente, d'où une réduction de la couverture. Cependant, ce phénomène semble disparaître lorsque la longueur maximale des séquences recherchées devient suffisamment grande.

À l'aide du graphique de la Figure 2.21, nous pouvons conclure que pour 10 mots-clés, un gradient de perplexité maximal de 1, un gradient de probabilité maximal de 0,1, un alpha

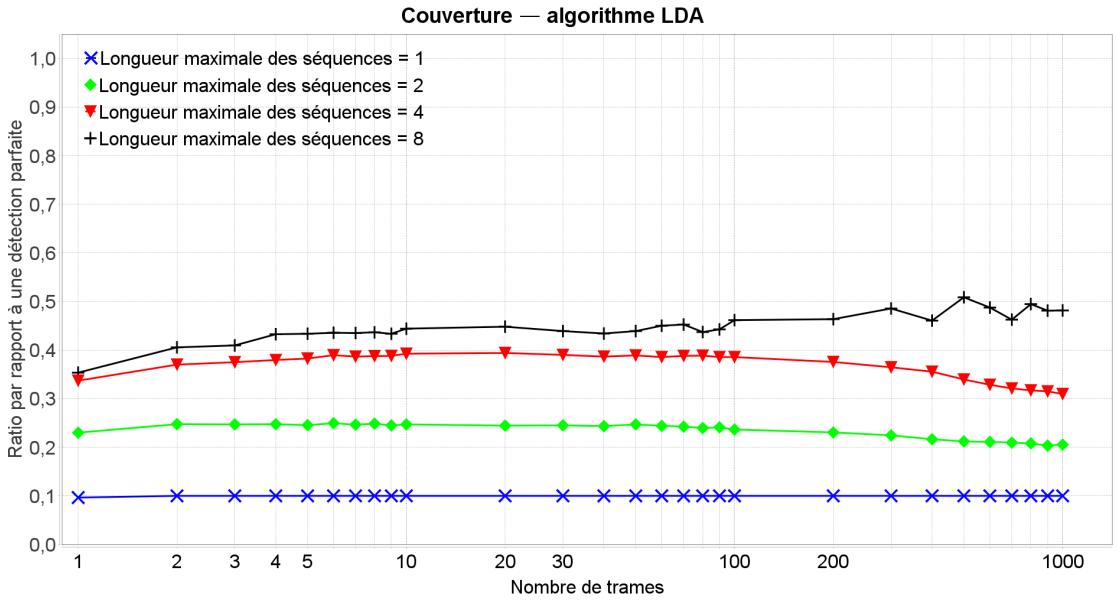


FIGURE 2.20 – Couverture de l'algorithme LDA en fonction du nombre de trames et paramétré par la longueur maximale des séquences

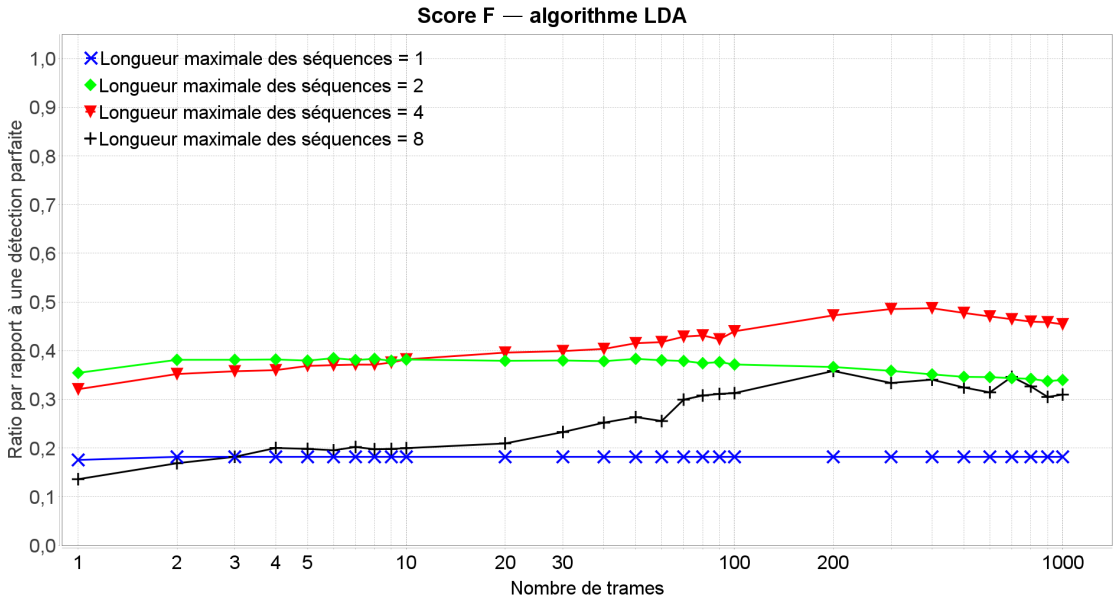


FIGURE 2.21 – Couverture de l'algorithme LDA en fonction du nombre de trames et paramétré par la longueur maximale des séquences

de 1 et un bêta de 0,0001, la longueur maximale des séquences recherchées maximisant les performances de l'algorithme LDA est de 4 bits.

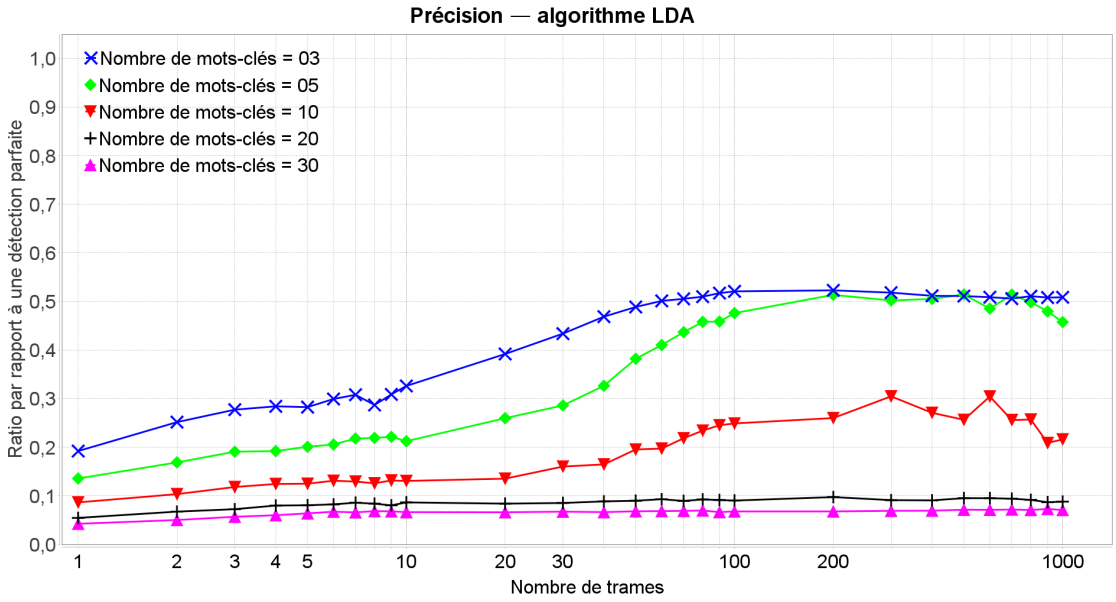


FIGURE 2.22 – Précision de l'algorithme LDA en fonction du nombre de trames et paramétré par son nombre de mots-clés

2.5.3 Influence du nombre de mots-clés recherchés

On observe sur le graphique de la Figure 2.22 que la précision de l'algorithme LDA décroit avec l'augmentation du nombre de mots-clés supposés présents dans la trace liaison de données. En effet, étant donné que l'on ne sélectionne que les séquences les plus probables pour chaque mot-clé, si l'on considère moins de mots-clés, alors on ne sélectionnera que les séquences les plus probables parmi celles qui auraient été sélectionnées s'il y avait eu plus de mots-clés. Ces séquences les plus probables sont généralement des séquences détectables, ce qui signifie plus de vrais positifs, et donc une plus grande précision.

On constate de plus, de façon générale, deux phases : une croissance linéaire par rapport à la taille de la trace liaison de données, pour la même raison que pour les courbes de précision paramétrées par la longueur maximale des séquences de la Figure 2.19 ; et une phase constante correspondant à la performance maximale atteignable avec la valeur fixée pour les autres paramètres.

Sur le graphique de la Figure 2.23, nous pouvons conclure que pour une longueur maximale de 8 des séquences recherchées, un gradient de perplexité maximal de 1, un gradient de probabilité maximal de 0,1, un alpha de 1 et un bêta de 0,0001, le nombre de mots-clés maximisant la performance de l'algorithme LDA sur l'intervalle simulé est de 3 ou 5. Cependant, nous privilégions les tailles de traces DLL plus grandes, donc nous choisissons 5.

À l'aide du graphique de la Figure 2.24, nous pouvons conclure que le nombre de mots-clés maximisant les performances de l'algorithme LDA sur l'intervalle simulé est de 3 ou 5. Toutefois, nous privilégions les tailles de trace liaison de données plus importantes, donc nous sélectionnons 5.

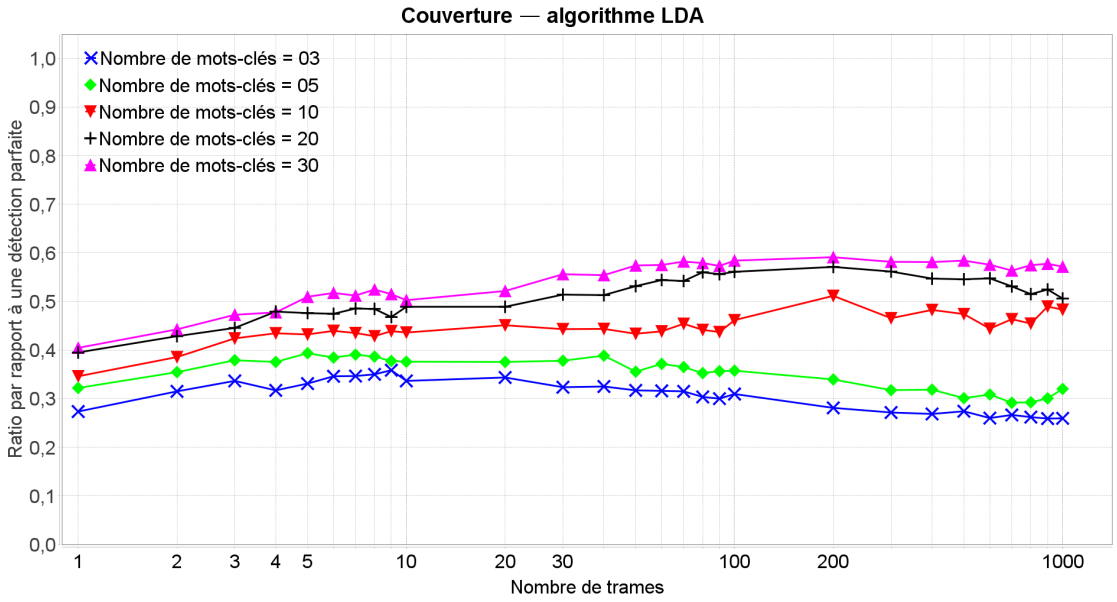


FIGURE 2.23 – Couverture de l'algorithme LDA en fonction du nombre de trames et paramétré par son nombre de mots-clés

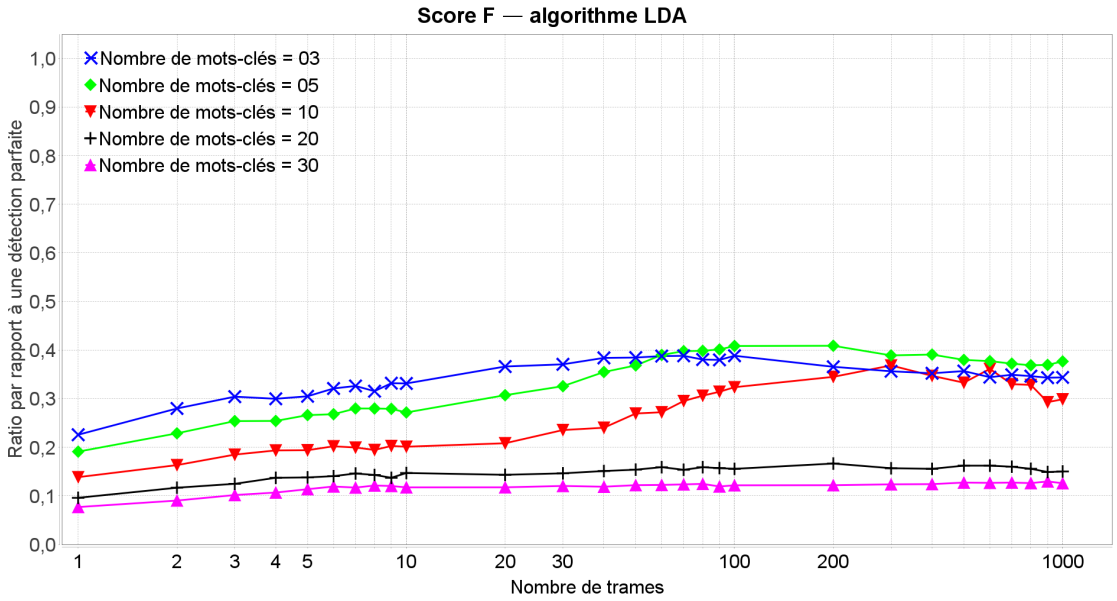


FIGURE 2.24 – Score F de l'algorithme LDA en fonction du nombre de trames et paramétré par son nombre de mots-clés

2.5.4 Influence du gradient maximal de perplexité avant arrêt de l'échantillonneur de Gibbs

On peut voir sur le graphique de la Figure 2.25 que la précision de l'algorithme LDA croit avec la valeur du gradient maximal de perplexité avant arrêt de l'échantillonneur de Gibbs, entre 0.3 et 3, et se stabilise ensuite. Ce qui signifie qu'itérer l'échantillonneur un grand nombre

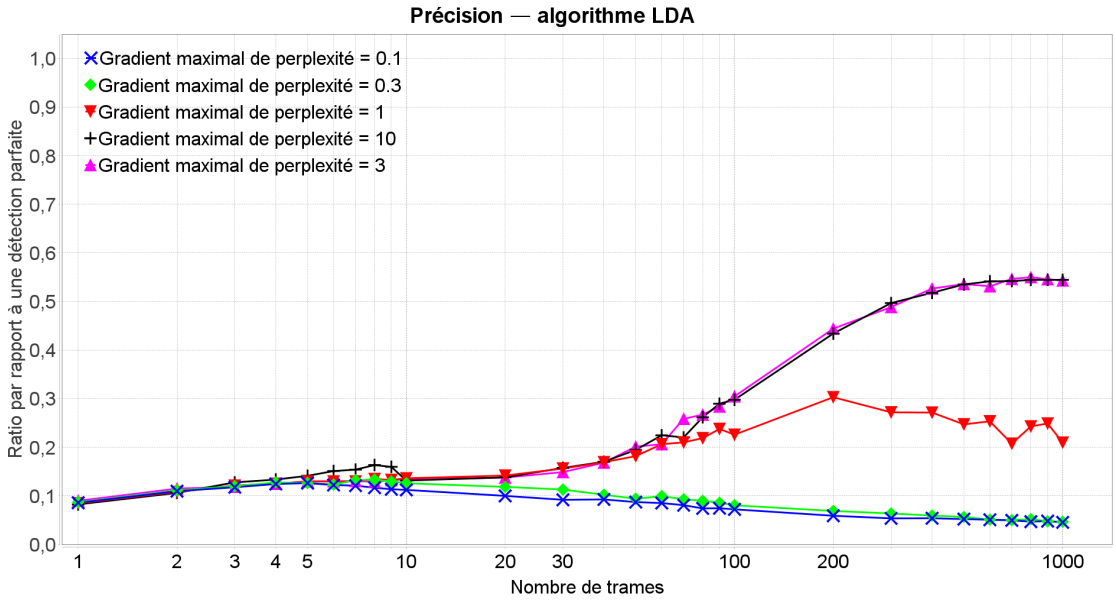


FIGURE 2.25 – Précision de l'algorithme LDA en fonction du nombre de trames et paramétré par son gradient maximal de perplexité

fois est inutile, et même à terme contre-productif, ce qui est très surprenant.

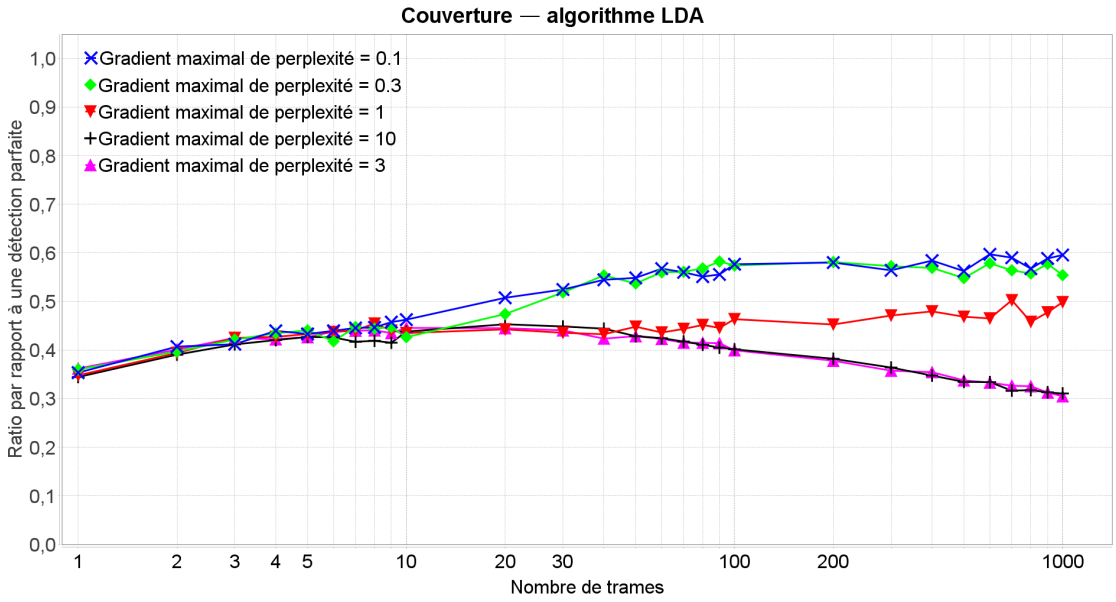


FIGURE 2.26 – Couverture de l'algorithme LDA en fonction du nombre de trames et paramétré par son gradient maximal de perplexité

On observe sur le graphique de la Figure 2.26 que la couverture de l'algorithme LDA décroit lorsque le gradient maximal de perplexité augmente. Cela signifie qu'itérer l'échantillonneur de Gibbs un plus grand nombre de fois permet de favoriser une plus grande diversité de mots-clés,

et donc de séquences retenues pour ces mots-clés.

Le comportement de la couverture relativement à la taille de la trace liaison de données est le même que pour les précédentes courbes de couverture de l'algorithme LDA (Figures 2.23 et 2.20), et pour la même raison.

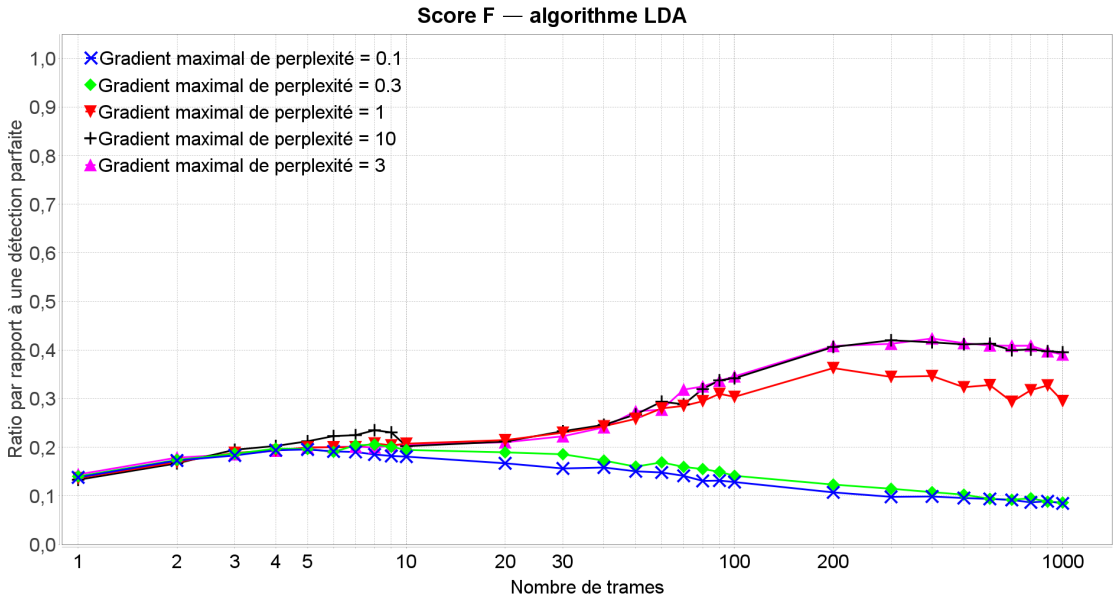


FIGURE 2.27 – Score F de l'algorithme LDA en fonction du nombre de trames et paramétré par son gradient maximal de perplexité

À l'aide du graphique de la Figure 2.27, nous pouvons conclure que pour 10 mots-clés, une longueur maximale de 8 des séquences recherchées, un gradient de probabilité maximal de 0,1, un alpha de 1, et un beta de 0,0001, les valeurs du gradient de perplexité maximal maximisant les performances de l'algorithme LDA sont celles supérieures ou égales à 3.

2.5.5 Influence du gradient maximal de probabilité pour la sélection des n-grams

On peut voir sur le graphique de la Figure 2.28 que la précision de l'algorithme LDA décroit avec l'augmentation du gradient maximal de probabilité. Cela paraît logique, car plus ce gradient est faible, plus les n-grams ayant la plus forte probabilité au sein de chaque mot-clé seront privilégiés. Hors ces derniers sont les n-grams étant le plus vraisemblablement des séquences remarquables du protocole.

On remarque également un comportement ressemblant dans l'ensemble à celui de la précision paramétrée par le nombre de mots-clés, avec une phase de croissance, puis une stabilisation. Le plafonnement est vraisemblablement dû au nombre de mots-clés trop important et au gradient de perplexité trop faible.

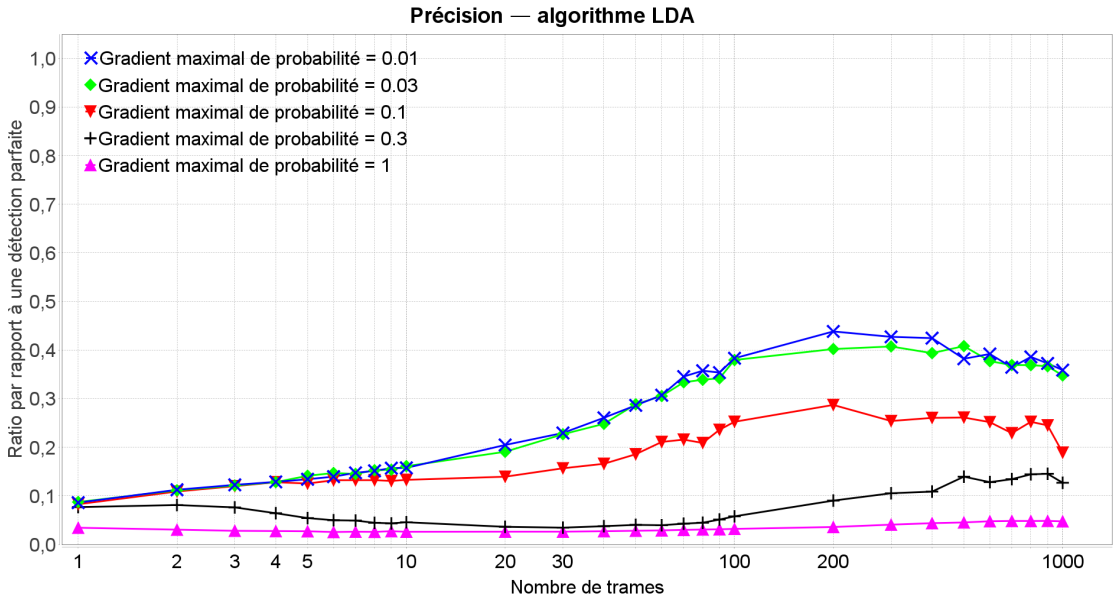


FIGURE 2.28 – Précision de l'algorithme LDA en fonction du nombre de trames et paramétré par son gradient maximal de probabilité

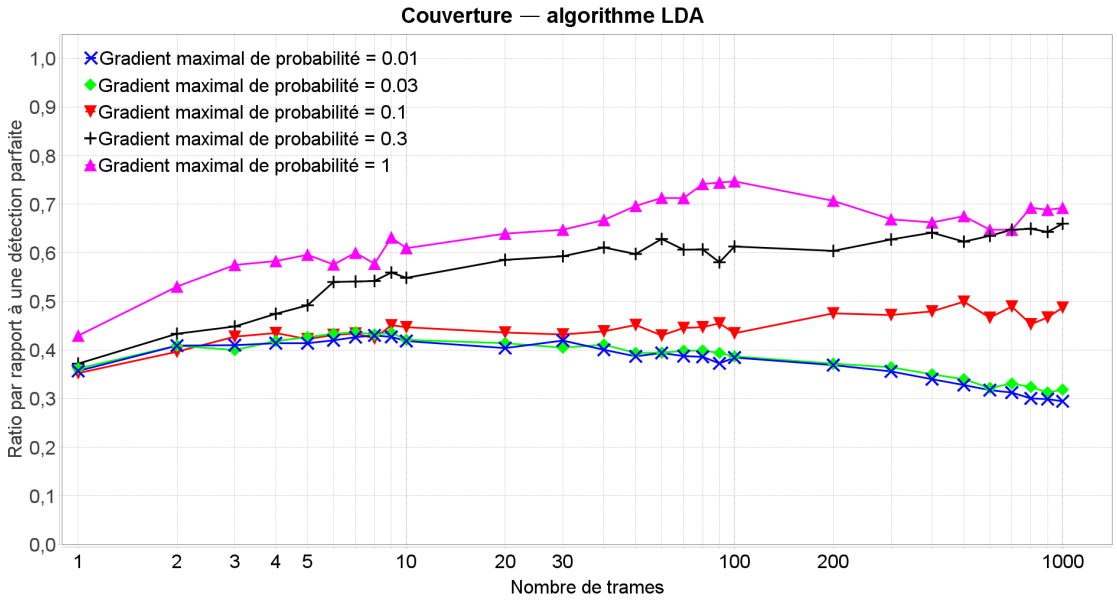


FIGURE 2.29 – Couverture de l'algorithme LDA en fonction du nombre de trames et paramétré par son gradient maximal de probabilité

On observe sur le graphique de la Figure 2.29 que la couverture de l'algorithme LDA croit lorsque le gradient maximal de probabilité augmente. Cela est dû au fait que lorsque ce gradient augmente, la dureté de la sélection des n-grams diminue. Par conséquent, un plus grand nombre est détecté, entraînant une augmentation de la diversité, et par conséquent de la couverture.

On constate un comportement relativement au nombre de trames similaire à celui de toutes les autres courbes de couvertures de l'algorithme LDA (Figures 2.23, 2.20 et 2.26), pour des raisons identiques.

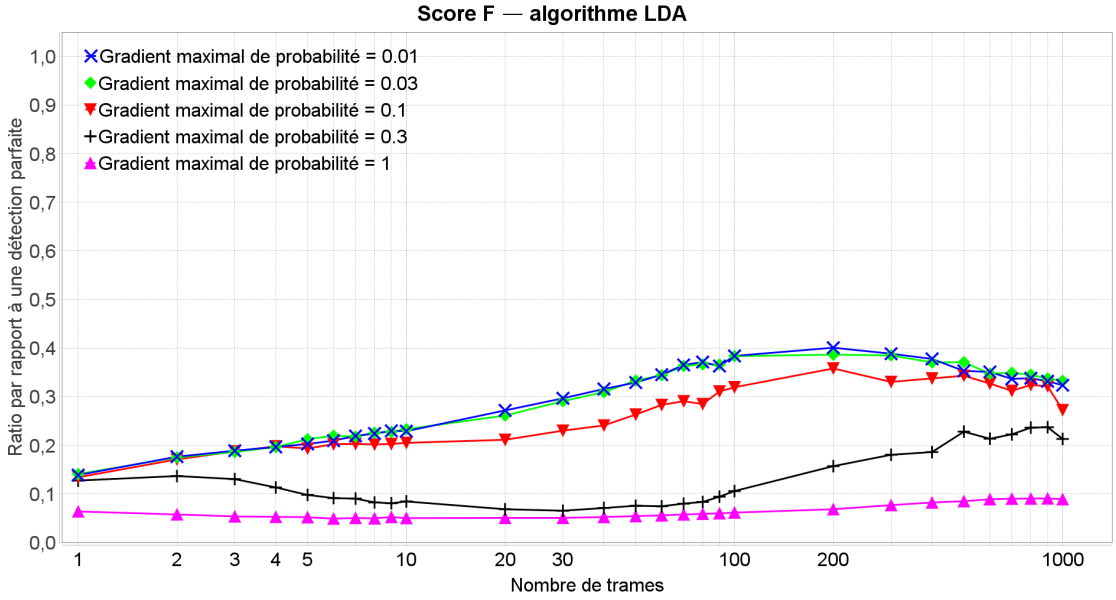


FIGURE 2.30 – Score F de l'algorithme LDA en fonction du nombre de trames et paramétré par son gradient maximal de probabilité

À l'aide du graphique de la Figure 2.30, nous pouvons conclure que pour 10 mots-clés, une longueur maximale des séquences recherchées de 8, un gradient de perplexité maximal de 1, un alpha de 1 et un bêta de 0,0001, la valeur du gradient de probabilité maximal maximisant la performance de l'algorithme LDA est de 0,03 ou moins.

2.5.6 Influence du paramètre 'alpha' de la distribution des mots-clés

Sur le graphique de la Figure 2.31, on observe que la précision de l'algorithme LDA croit avec alpha. En étudiant les caractéristiques de la loi de Dirichlet, on peut dire que cela signifie que l'algorithme offre sa meilleure précision lorsque l'on considère que les trames sont composées d'une mixture relativement homogène de tous les mots-clés du protocole, plutôt que de quelques uns seulement. On aurait plutôt attendu un comportement à l'opposé, à savoir que considérer seulement quelques mots-clés par trames favoriserait la précision.

On constate sur le graphique de la Figure 2.32 que la couverture est inversement décroit avec l'augmentation d'alpha. Ce comportement est là encore à l'opposé de ce que l'on aurait pu supposer, à savoir que considérer que les trames sont composées d'une mixture relativement homogène de tous les mots-clés du protocole améliorerait la couverture. Cependant, bien qu'inattendu, le comportement des courbes des deux graphiques reste cohérent avec le principe vérifié dans toutes les courbes précédentes (Figures 2.3 à 2.30), à savoir que précision et couverture présentent toujours des tendances opposées lorsque l'on fait varier un paramètre.

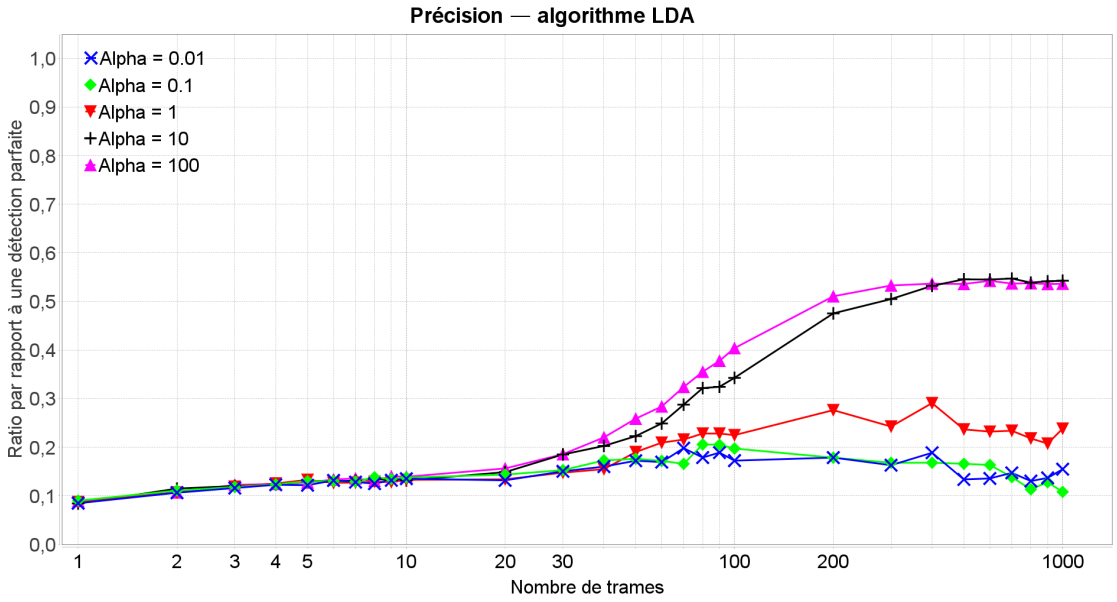


FIGURE 2.31 – Précision de l'algorithme LDA en fonction du nombre de trames et paramétré par son paramètre alpha

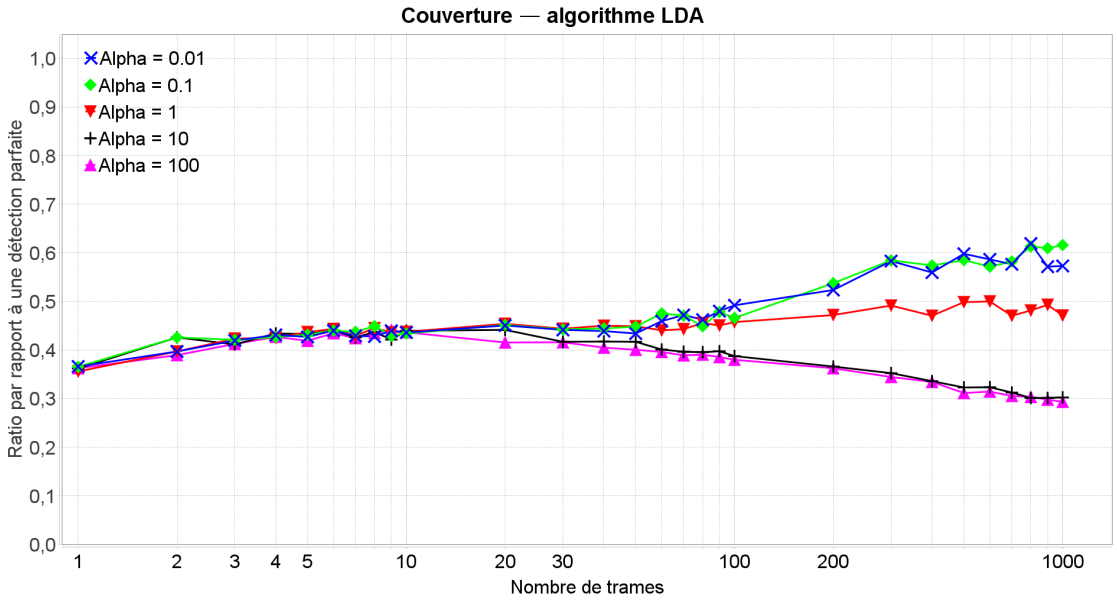


FIGURE 2.32 – Couverture de l'algorithme LDA en fonction du nombre de trames et paramétré par son paramètre alpha

Sur le graphique de la Figure 2.33 on peut voir que l'amélioration de la précision lorsque alpha augmente est plus importante que la détérioration de la couverture, ce qui résulte en une amélioration globale des performances. Ceci signifie que considérer que les trames sont composées d'une mixture relativement homogène de tous les mots-clés du protocole, ce qui est

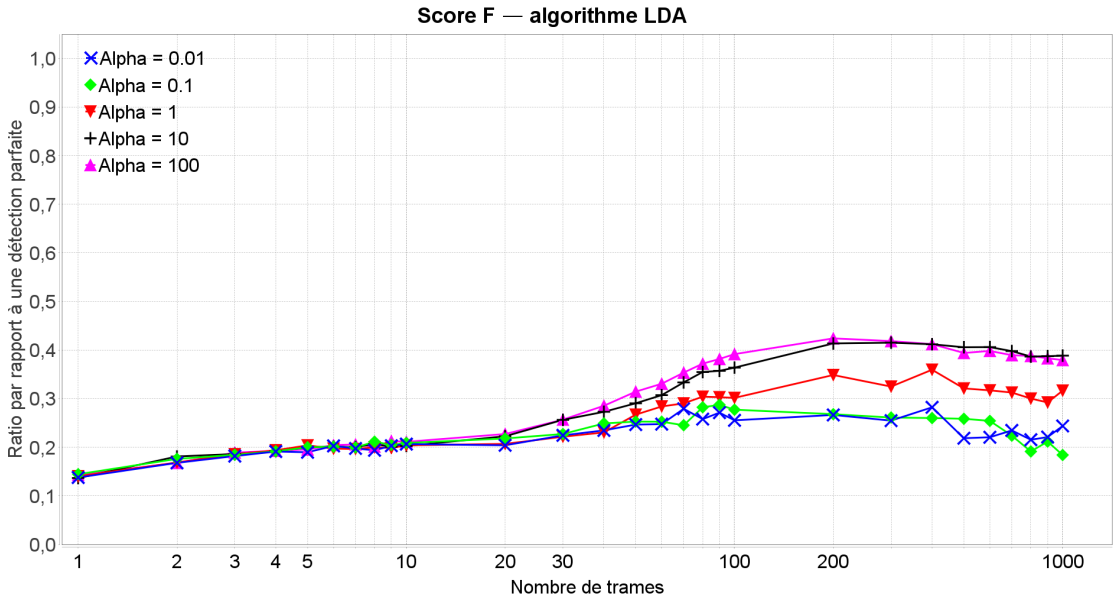


FIGURE 2.33 – Score F de l'algorithme LDA en fonction du nombre de trames et paramétré par son paramètre alpha

plus proche de la réalité que l'hypothèse inverse, conduit à une amélioration des performances, ce qui semble logique.

Nous pouvons donc conclure que pour 10 mots-clés, une longueur maximale de 8 des séquences recherchées, un gradient de perplexité maximal de 1, un gradient de probabilité maximal de 0,1 et un bêta de 0,0001, les valeurs d'alpha maximisant les performances de l'algorithme LDA sont celles supérieures ou égales à 10.

2.5.7 Influence du paramètre 'bêta' de la distribution des n-grams

Nous ne présentons pas les courbes de performance paramétrées par bêta car ce paramètre n'a pratiquement aucune influence compte tenu des valeurs de paramètres que nous avons simulées. Le paramètre bêta influence l'a priori sur la concentration de la distribution multinomiale des mots sur les sujets. Nous l'avons simulé pour des valeurs inférieures à 1, ce qui signifie qu'il a juste influencé l'importance de l'écart entre les valeurs de probabilité élevées et faibles. Cependant, avec la valeur maximale du gradient de probabilité que nous avons utilisée, les mots-clés sélectionnés sont toujours ceux qui se trouvent au-dessus de la séparation entre fortes et faibles probabilités, quelle que soit la largeur de la séparation.

2.6 Comparaison des performances des algorithmes et conclusion

Après avoir étudié indépendamment les performances de chacun des algorithmes, nous allons chercher à les comparer.

2.6.1 Paramètres des simulations

Nous avons dans un premier temps utilisé les valeurs optimales trouvées lors des simulations de l'influence de chacun des paramètres, mais les performances obtenues via cette méthode ont conduit à des performances sous-optimales. Nous avons donc décidé de sélectionner, pour chaque algorithme, la courbe de Score F présentant les meilleures performances, et avons récupéré le jeu de paramètres correspondant. Ceux retenus sont répertoriés dans la Table 2.5.

Paramètres	**Valeur des paramètres de la courbe retenue pour VDV**	**Valeur des paramètres de la courbe retenue pour AC**	**Valeur des paramètres de la courbe retenue pour LDA**
Nombre de trames	1 à 1000 par pas logarithmique	1 à 1000 par pas logarithmique	1 à 1000 par pas logarithmique
Longueur des séquences	4	4	4
Nombre de flux	10		
Seuil de filtrage de VDV	1		
Seuil de filtrage d'AC		1	
Seuil de fusion		1	
Nombre de mots-clés			10
Gradient maximal de perplexité			1
Gradient maximal de probabilité			0.1
Alpha			1
Bêta			0.0001

TABLE 2.5 – Courbes retenues pour la seconde comparaison

2.6.2 Discussion des résultats de la comparaison

On constate sur le graphique de la Figure 2.34 que pour les petites traces DLL (moins de 400 trames), l'algorithme AC offre la meilleure précision, suivi de l'algorithme LDA, et enfin de l'algorithme VDV. Pour les traces de plus de 400 trames, l'algorithme LDA devient plus précis que l'algorithme AC, et l'amélioration de la précision semble se poursuivre au-delà de 1000 trames.

Les performances maximales sont d'environ 85% de précision pour LDA, et un peu moins de 80% pour AC, tandis que VDV atteint un pic à un peu plus de 20%.

Sur le graphique de la Figure 2.35, on peut voir que l'algorithme LDA a la meilleure couverture quelle que soit la taille de la trace. Entre 10 et 100 trames, l'algorithme VDV a la deuxième meilleure couverture, et l'algorithme AC a la plus mauvaise, mais en dehors de ces

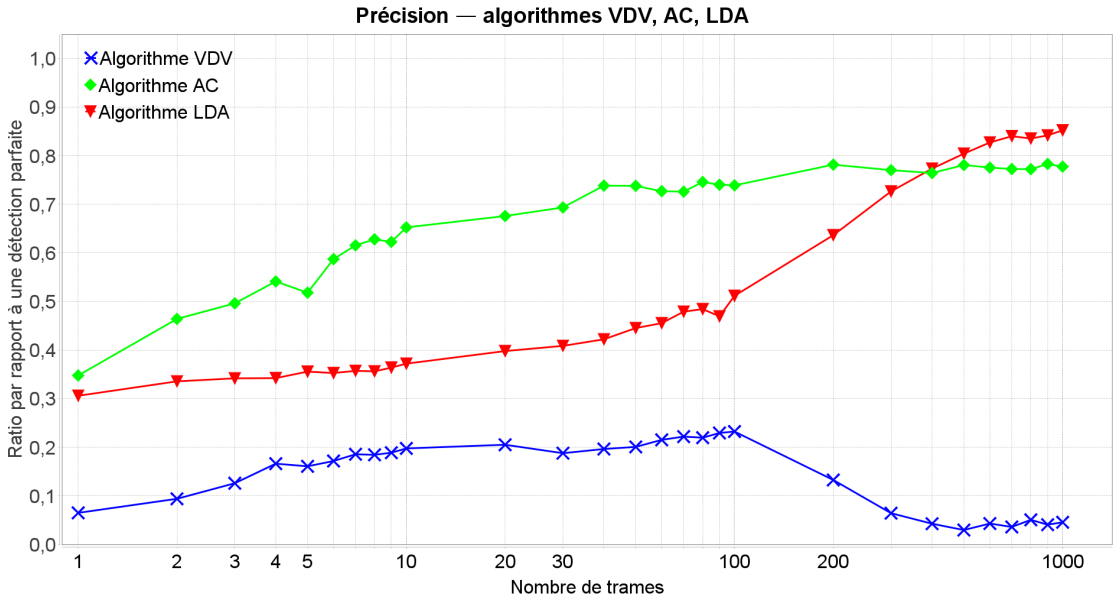


FIGURE 2.34 – Meilleure précision obtenue des algorithmes VDV, AC et LDA en fonction du nombre de trames

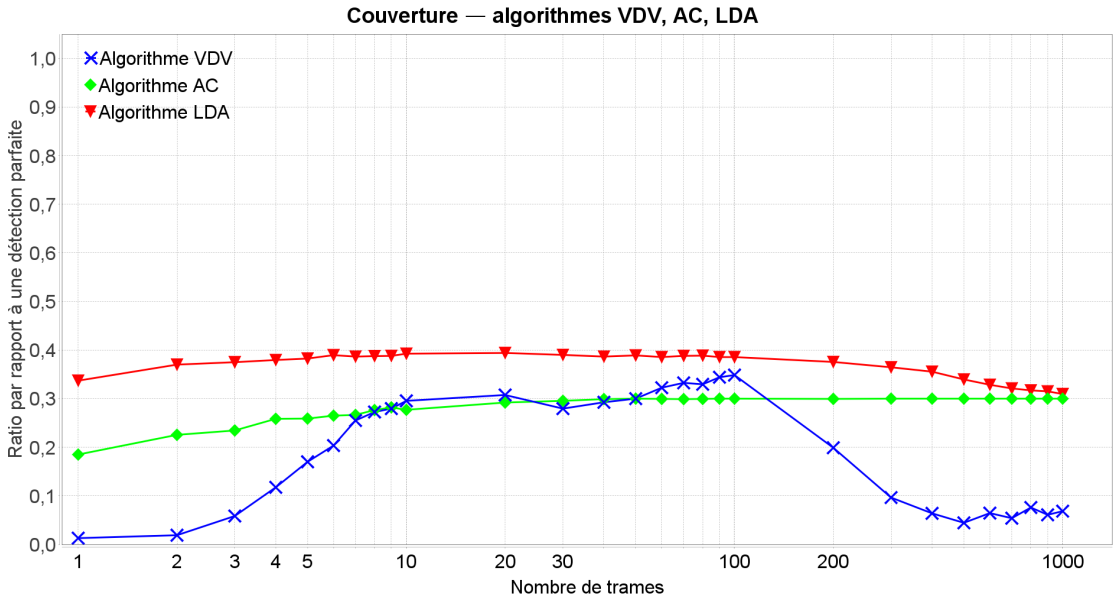


FIGURE 2.35 – Meilleure couverture obtenue des algorithmes VDV, AC, et LDA en fonction du nombre de trames

limites, AC est deuxième, et VDV dernier.

Les performances maximales sont d'environ 40% de couverture pour LDA, 35% pour VDV, et 30% pour AC.

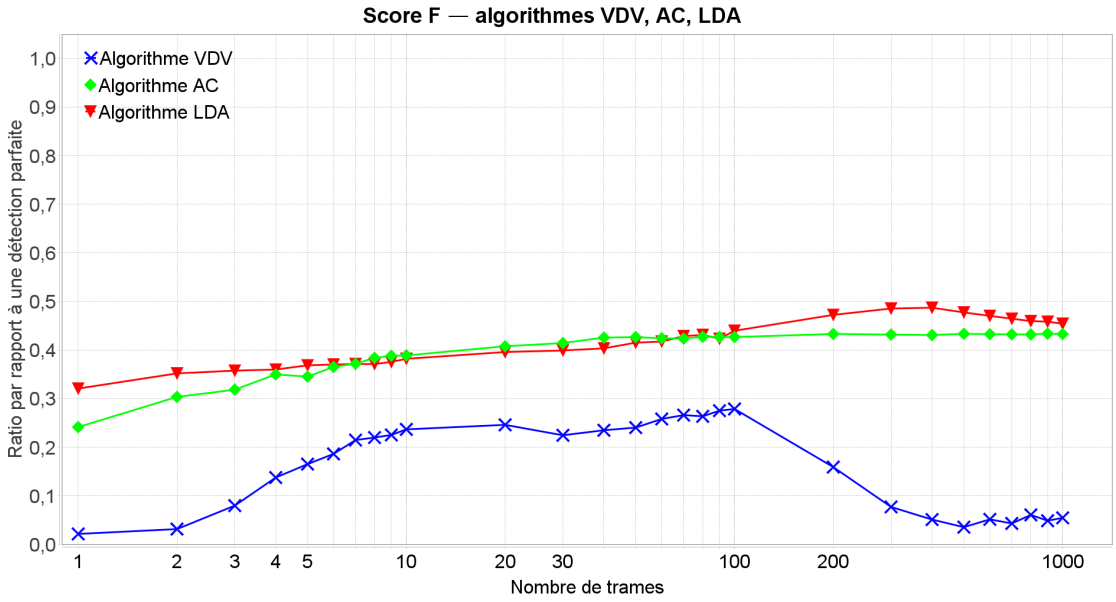


FIGURE 2.36 – Meilleur score F obtenu des algorithmes VDV, AC et LDA en fonction du nombre de trames

Le graphique de la Figure 2.36 montre que les algorithmes AC et LDA ont des performances approximativement identiques, le LDA semblant légèrement meilleur. L'algorithme VDV offre des résultats nettement moins bons, que même la détermination du seuil de filtrage en fonction de la taille de la trace DLL n'aurait pu compenser entièrement ; tout au plus, les résultats à 1000 trames auraient été du même ordre que ceux obtenus pour 100.

Les performances maximales sont légèrement inférieures à 50% de score F pour LDA, légèrement inférieure à 45% pour AC, et légèrement inférieure à 30% pour VDV.

En complément de l'évaluation de la pertinence des séquences détectées, nous évaluons maintenant la quantité d'informations détectées sur les champs. On obtient en conséquence les courbes suivantes :

On constate sur le graphique de la Figure 2.37 que le ratio moyen de détection des longueurs des champs de l'algorithme LDA est le meilleur, suivi de celui de AC, et enfin de VDV. Les performances maximales sont légèrement inférieures à 60% pour LDA, un peu moins de 50% pour AC, et légèrement inférieure à 30% pour VDV.

Le même comportement peut être observé sur les graphiques des Figures 2.38 et 2.39, concernant les ratios moyens de détection des valeurs des champs et des positions des champs. Leurs performances maximales sont, respectivement, de 75% pour l'algorithme LDA, légèrement supérieures à 40% pour l'AC, légèrement inférieures à 40% pour l'algorithme VDV, et légèrement inférieures à 60% pour l'algorithme LDA, légèrement inférieures à 50% pour l'AC, et légèrement inférieures à 10% pour l'algorithme VDV.

On présente dans la Table 2.6 un bilan des performances comparées des algorithmes basé

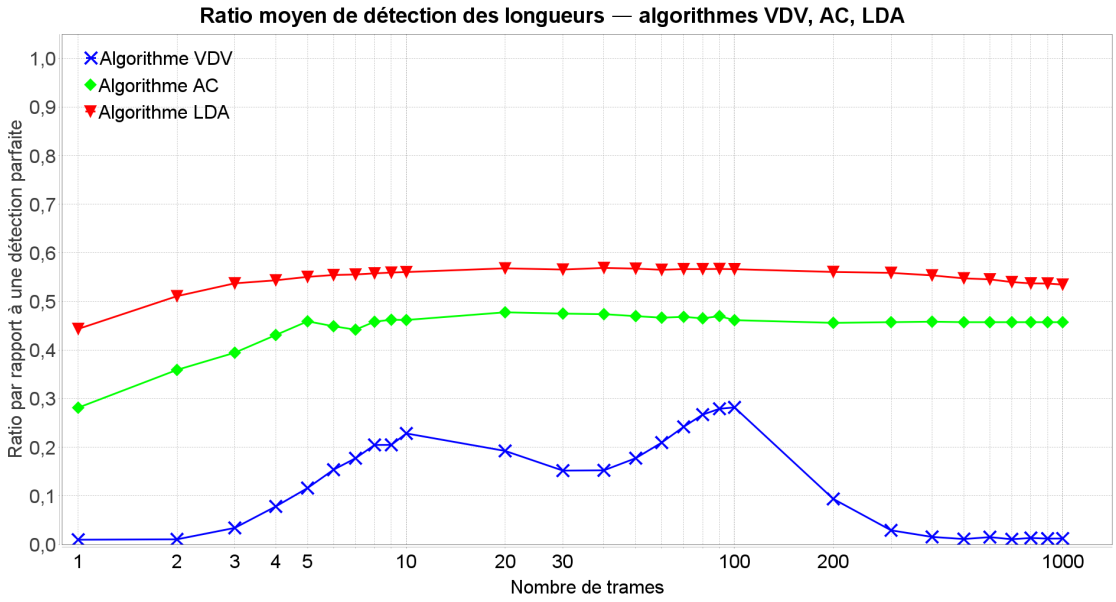


FIGURE 2.37 – Meilleur ratio moyen de détection des longueurs obtenu des algorithmes VDV, AC et LDA en fonction du nombre de trames

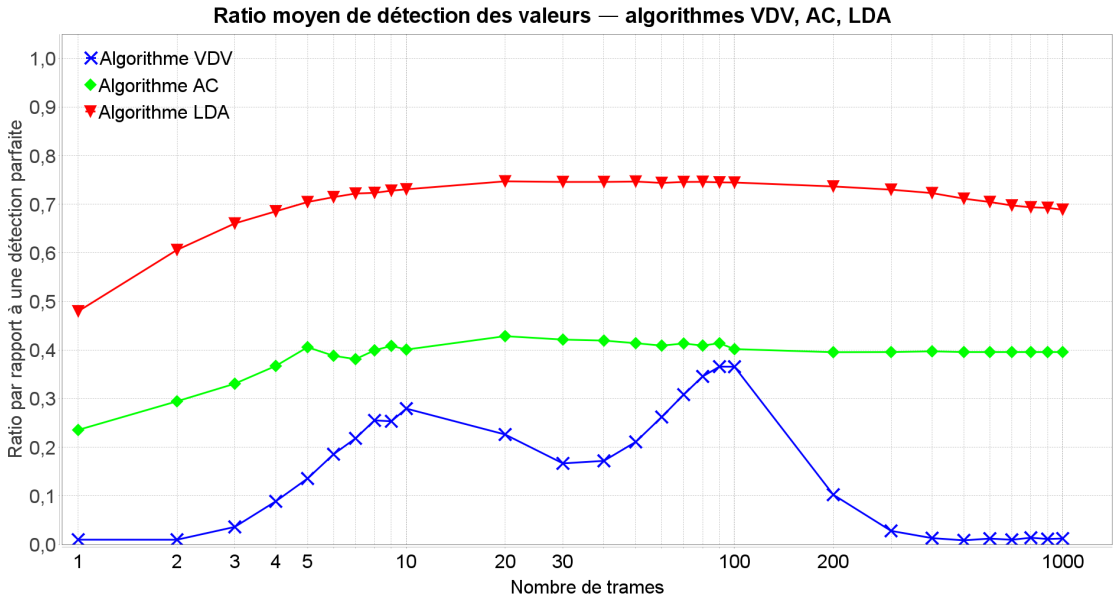


FIGURE 2.38 – Meilleur ratio moyen de détection des valeurs obtenu des algorithmes VDV, AC et LDA en fonction du nombre de trames

sur les courbes de simulations présentées dans cette comparaison. On y voit très clairement que les algorithmes VDV et AC sont largement inférieurs à l'algorithme LDA. De plus, ce dernier semble présenter des perspectives d'amélioration de performances si l'on venait à augmenter la taille de la trace liaison de données. Dans notre configuration d'étude, l'algorithme LDA est donc le meilleur des trois algorithmes simulés.

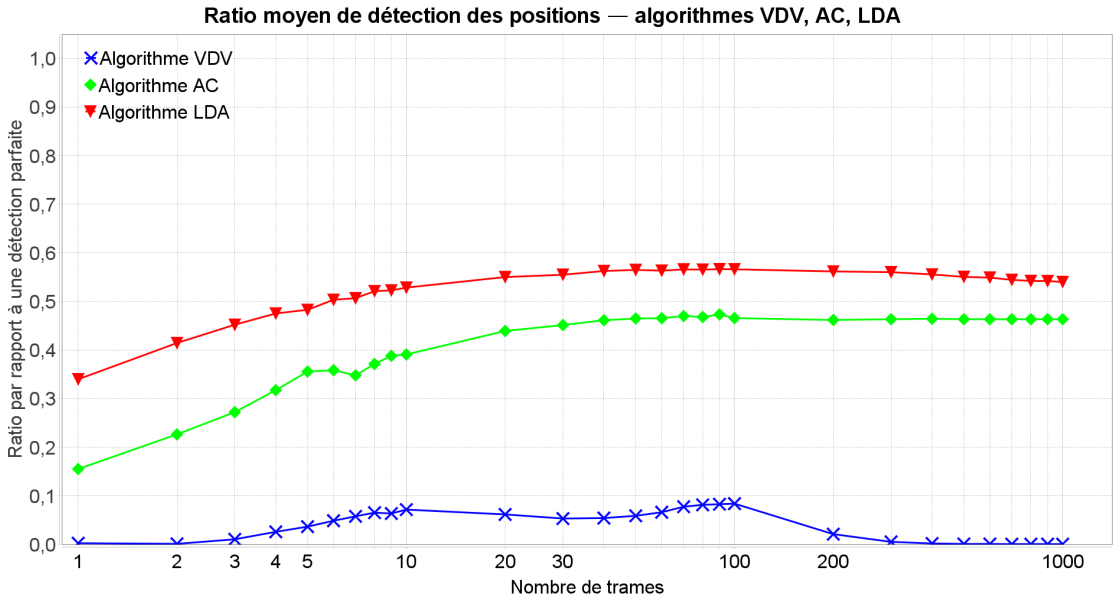


FIGURE 2.39 – Meilleur ratio moyen de détection des positions obtenu des algorithmes VDV, AC et LDA en fonction du nombre de trames

Algorithmes	VDV	AC	LDA
Meilleure précision	*	***	***
Meilleure couverture	*	**	***
Meilleur score F	*	***	***
Meilleur ratio moyen de détection des longueurs	*	**	***
Meilleur ratio moyen de détection des valeurs	*	**	***
Meilleur ratio moyen de détection des positions	*	**	***
Total	6	14	18

TABLE 2.6 – Bilan des performances des algorithmes VDV, AC et LDA

2.7 Conclusion

Après avoir étudié théoriquement les trois techniques d'identification de séquences, nous les avons simulés afin d'en comparer les performances. Pour cela, nous avons utilisé les métriques classiques de précision, couverture, et score F, ainsi que notre propre métrique, le ratio de détection des champs

Afin de comparer les meilleures performances de chaque algorithme, nous avons dans un premier temps étudié l'influence de chacun de leurs paramètres, afin de trouver la meilleure combinaison. Cette démarche est simple à mettre en œuvre, mais toutefois assez naïve ; nous ne pouvons pas parler de résultats optimaux atteignables avec ces algorithmes. De plus, pour des raisons matérielles, la taille des traces liaison de données simulées est restée assez limitée.

Malgré cela, nous considérons que les résultats obtenus sont représentatifs des performances relatives des différentes techniques.

Suite à cette première étude, nous avons constaté que les performances de l'algorithme VDV ne sont pas constantes en fonction du nombre de trames présentes dans la trace. En effet, elles chutent passé 100 trames, car le seuil de filtrage ne s'adapte pas à la taille de la trace analysée. Le score F maximal obtenu avec cette algorithme atteint à peine les 30%.

Concernant les performances de LDA et AC, nous avons pu constater qu'ils offraient un comportement assez proche, augmentant lentement avec le nombre de trames dans la trace, avant de plafonner aux alentours de 1000 trames. Sur le plan du score F, le LDA ne devance l'AC que de quelques pourcents, avec un peu plus de 45% contre un peu moins de 45%.

En prenant en compte la totalité des métriques considérées, nous avons pu conclure que l'algorithme LDA est sensiblement plus performant que l'algorithme AC, lui même largement plus performant que l'algorithme VDV.

Cette conclusion nous encourage à nous tourner vers le domaine des réseaux Bayésiens, dont est issu l'algorithme LDA. Nous allons donc, dans le prochain chapitre, introduire la théorie des réseaux Bayésiens, puis présenter notre propre modèle, BaNet3F, basé sur un de ces réseaux.

Chapitre 3

Proposition d'un nouveau modèle : BaNet3F

Dans ce chapitre nous allons commencer par exposer les éléments essentiels de la théorie des réseaux Bayésiens, à savoir leur principes fondamentaux, leur représentation, et comment effectuer les procédures d'inférence exacte et d'apprentissage. Nous nous intéresserons également aux réseaux Bayésiens dynamiques, car notre modèle est un croisement entre ces deux types de réseaux. L'essentiel de cette partie est basé sur la thèse de K. P. Murphy [67].

Nous présenterons ensuite les principales contributions de cette thèse, c'est-à-dire le modèle d'apprentissage de formats de trames binaires proposé, ainsi qu'une adaptation de l'algorithme de Viterbi aux réseaux Bayésiens quelconques.

Il est à noter que dans toute cette thèse, seul le cas discret sera présenté et utilisé, car nous traitons de communications numériques, donc discrètes par définition. D'un point de vue pratique, les distributions de probabilités seront donc toujours des tables, et non des fonctions.

3.1 Réseaux Bayésiens

3.1.1 Principes fondamentaux et représentation des réseaux Bayésiens

Les réseaux Bayésiens (RBs) sont une famille de modèles graphiques permettant de représenter un système à travers les liens probabilistes entre ses différentes variables caractéristiques. Les nœuds du graphe représentent les variables, tandis que les arcs orientés représentent les liens conditionnels entre variables. L'intérêt de ce réseau réside dans le fait qu'il donne la possibilité d'inférer à faible coût n'importe quelle relation probabiliste entre les variables.

Dans un réseau Bayésien, n'importe quelle relation conditionnelle est possible tant qu'aucune relation cyclique n'apparaît, c'est à dire qu'aucune variable ne dépend directement ou indirectement d'elle-même.

La construction d'un tel réseau se déroule en quatre étapes :

1. Dans un premier temps, on identifie les variables aléatoires caractéristiques de notre système. Plus le nombre de variables est important, plus le système peut contenir d'in-

formations, mais sa complexité augmente linéairement.

2. Suite à quoi, on définit le nombre d'états possibles pour chaque variable. Cela permet à une variable de contenir plus d'informations, mais augmente linéairement la complexité du système.
3. Vient ensuite la spécification des liens conditionnels entre variables. Cette étape, à la différence des deux précédentes, peut être effectuée de deux manières : manuellement, si l'on connaît a priori les relations entre variables, comme c'est le cas dans le cadre de cette thèse, ou alors automatiquement, par apprentissage, si ce n'est pas le cas. Cet apprentissage n'est pas présenté dans le cadre de cette thèse, mais un algorithme classique permettant de l'effectuer est une variante de l'*Expectation-Maximization*, que nous allons voir plus loin dans la section 3.1.4, appelée *Structural Expectation-Maximization* [67]. Augmenter le nombre de liens présents dans le réseau permet d'inclure plus de subtilité dans les dépendances entre variables, mais fait augmenter exponentiellement la complexité du réseau. Il s'agit donc généralement du principal levier permettant d'ajuster le compromis entre performances et volume de calculs à effectuer.
4. Enfin, la dernière étape consiste à préciser les paramètres exacts des tables de probabilités de chacune des variables. Généralement, et c'est le cas dans cette thèse, tout ou partie des distributions des variables est inconnu, et c'est pourquoi un apprentissage est mis en place afin de trouver des paramètres permettant d'expliquer au mieux les observations du système.

Une fois le réseau construit, il devient possible d'y appliquer divers algorithmes pour effectuer l'opération d'inférence. Les résultats de l'inférence permettent ensuite de calculer simplement n'importe quelle probabilité souhaitée. Nous verrons cette procédure dans la section 3.1.2.

Il existe trois types de variables dans un RB :
— Les **variables d'entrée**, qui sont maîtrisées et observables
— Les **variables de sortie**, qui ne sont pas maîtrisées, mais sont observables
— Les **variables cachées**, qui ne sont ni maîtrisées, ni observables

Prenons l'exemple du réseau représenté sur le schéma de la Figure 3.1. Ce réseau comporte quatre variables : a, b, c, et d. La variable a est une variable d'entrée, c'est à dire qu'il s'agit d'un paramètre du système sur lequel l'utilisateur a la main. a peut prendre 3 états possibles : a_0, a_1, et a_2. Les variables b et c sont cachées et peuvent prendre deux états : b_0 et b_1, c_0 et c_1, respectivement. d est une variable de sortie, c'est à dire qu'on observe sa valeur sans pouvoir influer directement dessus. d peut prendre deux états : d_0 et d_1. Toutes les variables sont conditionnellement dépendantes les unes des autres, directement ou indirectement, mais les seules distributions de probabilités explicitées dans le réseau sont celles de b, conditionnellement dépendante à a, et d, conditionnellement dépendante à b et c. D'un point de vue graphique, on peut dire que b est enfant de a ou a parent de b, et que d est enfant de b et c, ou b et c sont parents de d. Cela signifie que les distributions de probabilités explicitement incluses dans le réseau sont : P(a), P(b|a), P(c), et P(d|b,c). Une représentation possible des tables de probabilités est présente à côté de chacun des nœuds.

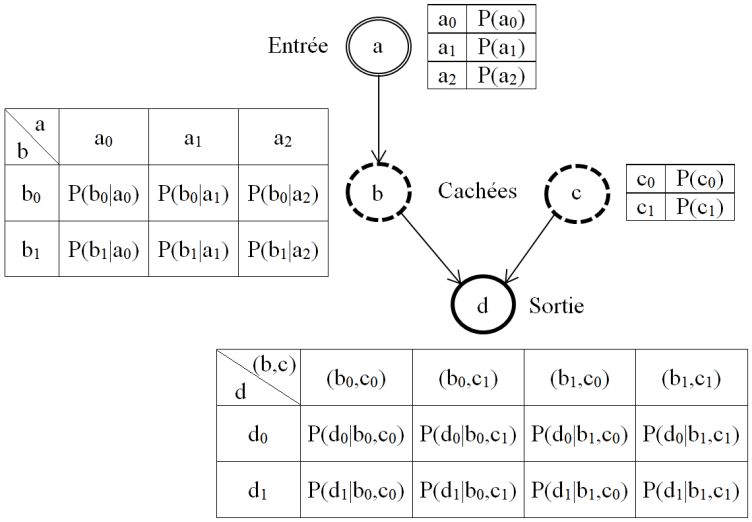


FIGURE 3.1 – Un exemple de réseau Bayésien

La probabilité conjointe de toutes les variables, à partir de laquelle on peut extraire n'importe quelle probabilité désirée peut être décomposée comme suit :

$$P(a, b, c, d) = P(d|b, c) \times P(c) \times P(b|a) \times P(a) \tag{3.1}$$

Comme on peut le voir, la probabilité conjointe peut être calculée sans avoir besoin de sa table explicite. Le nombre de paramètres nécessaires est donc de $8 + 2 + 6 + 2 = 18$ avec les tables de chacun des nœuds contre $3 \times 2 \times 2 \times 2 = 24$ avec la table conjointe des deux variables. Un réseau Bayésien permet donc de transformer un volume exponentiel des tables en un volume linéaire. De plus, concernant la marginalisation d'une variable, d par exemple, on a :

$$P(d) = \sum_{a,b,c} P(a, b, c, d) = \sum_{a} P(a) \times \sum_{b} P(b|a) \times \sum_{c} P(c) \times P(d|b, c) \tag{3.2}$$

Il ne s'agit là que d'une décomposition possible de la somme, certaines étant meilleurs que d'autres, mais ce point est à lui seul un domaine à part entière, auquel nous ne nous intéresserons pas dans le cadre de cette thèse. La somme décomposée comporte $2 + 1 + 4 + 1 + 2 + 1 + 4 = 15$ opérations, tandis que la somme non décomposée en comporte $3 \times 2 \times 2 = 12$. On constate donc que, si l'on ignore le nombre d'opérations en lui-même, peu représentatif sur notre exemple, la complexité croit d'un côté linéairement, et de l'autre exponentiellement. Cette propriété permet d'effectuer l'inférence sur de larges réseaux sans explosion du volume de calculs nécessaire.

Ceci est permis grâce à l'exploitation de deux propriétés fondamentales des réseaux Bayésiens : la réduction des dépendances et l'indépendance conditionnelle. Par réduction des dépendances, on entend qu'au lieu de considérer chaque variable dépendante de toutes les autres, on considère la majorité des liens entre variables nuls ou négligeables, afin de se ramener à un cas où une décomposition de complexité linéaire des probabilités devient possible. L'indépendance entre deux variables conditionnellement à une troisième, par exemple a et d conditionnellement à b dans notre réseau, se traduit par le fait que si l'état de b est connu, alors l'état de

a n'a aucune influence sur celui de d et vice versa ; cette propriété permet de considérer indépendamment les variables (a,b) et (b,d), et donc de réduire la complexité des calculs. Cette indépendance conditionnelle se formule mathématiquement comme suit :

$$a \perp d|b \tag{3.3}$$

3.1.2 Inférence exacte dans un réseau Bayésien

L'objectif de l'inférence est de trouver un ensemble de distributions de probabilités permettant ensuite, par marginalisation et produits de déterminer n'importe quelle probabilité du système. Cela se traduit de façon pratique par la détermination d'un ensemble de distributions de probabilités conjointes des variables du système à partir des observations effectuées sur certaines variables, et des distributions de probabilités conditionnelles et marginales connues du système. Les tables de probabilités ainsi obtenues contiennent donc toute l'information nécessaire du réseau tout en restant de taille raisonnable (pour peu que le nombre de liens conditionnels entre variables soit bien choisi).

L'inférence que nous présentons ici est dite exacte car elle vise à trouver les distributions de probabilités conjointes exactes des variables, par opposition à une inférence approximée, qui ne cherche qu'à s'en approcher.

La **technique d'inférence exacte** présentée ici est basée sur l'**algorithme de *Message Passing***, comme expliqué par K. P. Murphy dans [67]. Cet algorithme n'est applicable au réseau que lorsque celui-ci est mis sous la forme d'un ***junction tree***, que nous allons présenter un peu plus loin.

Le schéma de la Figure 3.2 donne une vue d'ensemble de la procédure complète d'inférence par *Message Passing*. Celle-ci se déroule en cinq étapes :

1. Construction du *junction tree* :
   (a) Moralisation du graphe
   (b) Triangularisation du graphe
   (c) Procédure d'élimination des variables
   (d) Assemblage du junction tree
2. Exécution du *Message Passing*

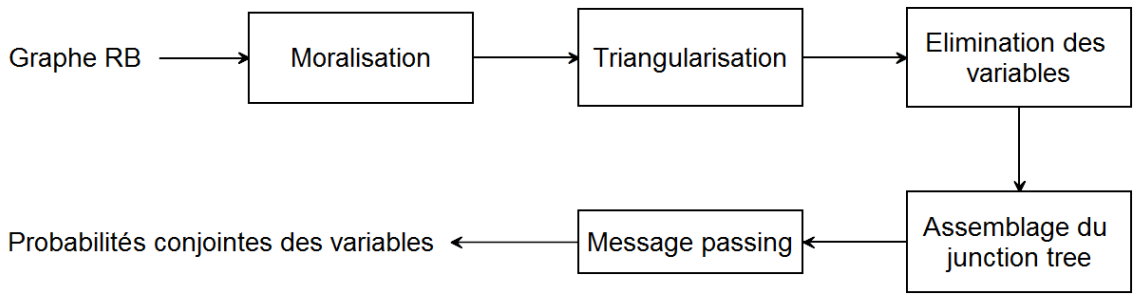
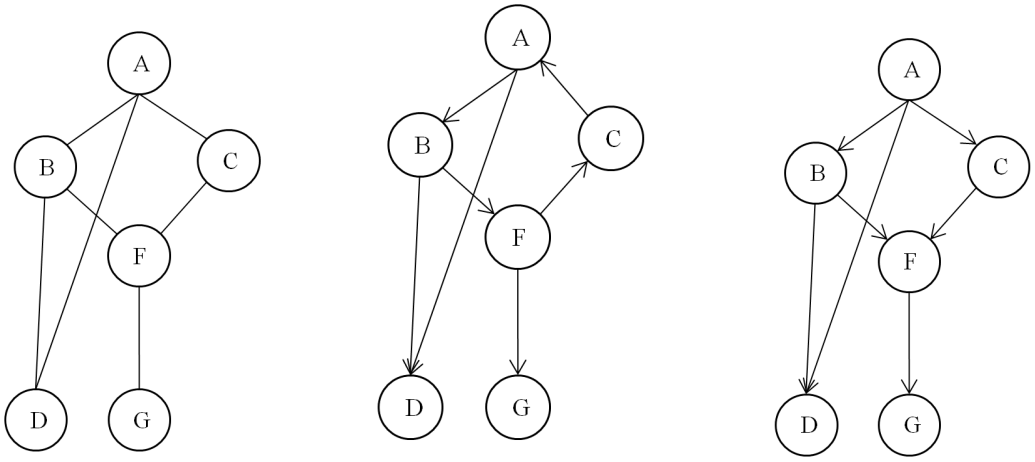


FIGURE 3.2 – Procédure d'inférence exacte dans un réseau Bayésien via *Message Passing*

Construction du *junction tree*

Les quatre premières étapes ont pour but de transformer le graphe du réseau en *junction tree*. Un *junction tree* est un arbre vérifiant la Running Intersection Property (RIP), qui sera présentée un peu plus loin, et dont les nœuds sont des sous-graphes fortement connexes du graphe de départ. Chacun de ces nœuds est appelé une clique et rassemble un groupe de variables du réseau, toutes connectées deux à deux. Le *junction tree* permet de simplifier un graphe en faisant disparaître les cycles non directs tout en conservant ses propriétés, ce qui rend possible l'utilisation de l'algorithme de *Message Passing*. Cela permet de réduire fortement la complexité du mécanisme d'inférence.

Le graphe de départ est un Graphe Orienté Acyclique (GOA), c'est à dire un graphe dont tous les arcs sont orientés, et ne présentant aucun cycle. Les Figures 3.3a , 3.3b ,et 3.3c présentent, respectivement, un graphe non-orienté, un graphe orienté non-acyclique, et un GOA. Ce dernier graphe servira d'exemple tout au long de ce chapitre.



(a) Un graphe non-orienté (b) Un graphe orienté non-acyclique (c) Un GOA

FIGURE 3.3 – Trois exemples de graphes

Moralisation : La première opération effectuée sur le GOA est la moralisation, consistant à lier entre eux tous les parents d'un même nœud à l'aide d'un arc bidirectionnel, et à supprimer la directionnalité des arcs. Cette opération permet de s'assurer de la forte connexité de chaque sous-ensemble constitué par un nœud et ses parents. Le schéma de la Figure 3.4 présente en pointillé l'arc ajouté.

Triangularisation : L'étape suivante consiste à trianguliariser le graphe moralisé, c'est à dire à ajouter des arcs de façon à s'assurer qu'il n'existe aucun cycle ininterrompu de longueur supérieure ou égale à 4. Les schémas des Figures 3.5a et 3.5b présentent respectivement un graphe non-direct non triangularisé, et un graphe non-direct triangularisé. Dans le cas de l'exemple choisi, il se trouve que la triangularisation n'est pas nécessaire, car la moralisation le triangularise naturellement.

Il est à noter que, contrairement à la moralisation, il existe généralement plusieurs possi-

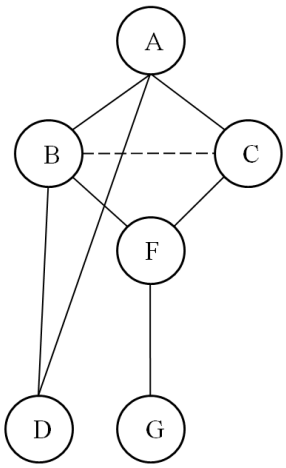
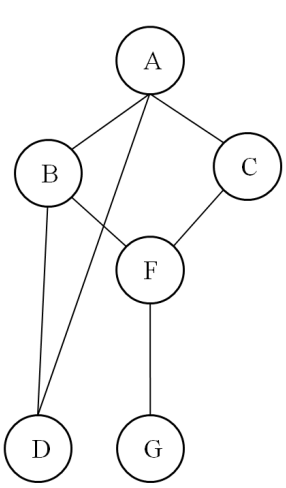
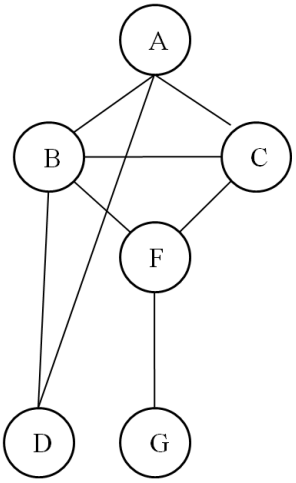


FIGURE 3.4 – Graphe de la Figure 3.3c moralisé

bilités de placement des arcs pour triangulariser un graphe.



(a) Graphe non-direct non triangularisé



(b) Graphe de la Figure 3.4 triangularisé (identique au moralisé)

FIGURE 3.5 – Exemple de triangularisation

Elimination des variables : À partir du graphe triangularisé, on cherche ensuite à effectuer l'élimination des variables, c'est à dire à rassembler les nœuds du graphe en cliques à l'aide d'un ordre d'élimination déterminé. Ces cliques constituent les nœuds du *junction tree*, et sont des ensembles de nœuds du graphe original, ayant été rendus fortement connexes par triangularisation et moralisation. Par exemple, $\{B, C, F, G\}$ pourrait être une clique. Il faut cependant noter que les cliques constituées dépendent entièrement de l'ordre d'élimination, qui n'est pas unique. Il en existe une multitude possibles, et leur choix impacte fortement sur l'efficacité de l'élimination des variables. Si le graphe est triangularisé, ce qui est le cas grâce

à la procédure que nous avons précédemment présentée, alors il existe un ordre d'élimination dit parfait, c'est à dire permettant d'obtenir le moins de cliques possibles et les plus petites possibles.

La procédure de formation des cliques est la suivante :
— On considère chaque nœud l'un après l'autre dans l'ordre d'élimination
— On forme une clique avec ce nœud et tous ses voisins, en rajoutant les arcs nécessaires pour avoir une forte connexité (si l'ordre d'élimination est bien choisi, il n'y en a normalement pas ou peu besoin)
— On élimine du graphe le nœud considéré

Cette procédure est formalisée dans l'Algorithme 5.

Algorithme 5 : Algorithme d'élimination des variables

Données : ordre d'élimination π, Graphe G
Résultat : Ensemble des cliques $\{C_i\}_{1 \leq i \leq N}$
pour $i = N$ *à* 1 *par pas de* -1 **faire**
  nœud $V_i = \pi(i)$;
  Former C_i à partir de V_i et tous ses voisins non-éliminés;
  Rajouter les arcs nécessaire afin que C_i soit fortement connexe;
  Éliminer V_i;
fin

La série de Figures de 3.6a à 3.6f illustre le déroulement de cet algorithme itération par itération, pour l'ordre d'éliminatrion $\pi = \{A, B, C, D, F, G\}$.

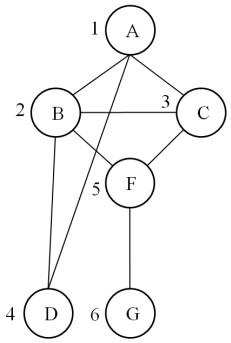
Assemblage du *junction tree* : Désormais, le graphe a été découpé en cliques, qu'il va maintenant falloir assembler pour constituer le *junction tree*. Ce dernier utilise en effet pour nœuds les cliques créées. L'une de ses propriétés les plus importantes est que, pour n'importe quelle paire de ses nœuds, leur intersection (les variables du réseau d'origine en commun) est présente tout le long de l'unique chemin les reliant.

Ceci n'est possible que si la Running Intersection Property est vérifiée, c'est à dire que, pour tout C_j, il existe un C_i, i < j, tel que $C_j \cap \cup_{1 \leq k \leq j-1} C_k \subseteq C_i$. Par exemple $C_4 \cap \{C_1, C_2, C_3\} = \{A, B, D\} \cap \{A, B, C\} = \{A, B\} \subseteq C_2$. Dans le cas de notre exemple, cette propriété est vérifiée.

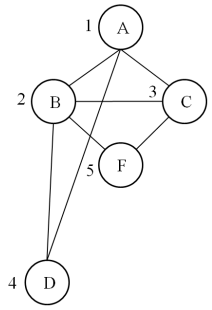
On supprime généralement aussi toute redondance, afin qu'il n'existe pas de couple $(C_i,\ C_j)$ tel que $C_i \subseteq C_j$. Dans notre exemple, on supprime ainsi les cliques C_2 et C_1. Il est important de vérifier que la RIP est toujours vérifiée après cette suppression. C'est le cas dans notre exemple car il s'agit des dernières cliques créées, mais lorsque les doublons sont au milieu de l'ordre de création des cliques, ce n'est pas forcément le cas.

Nous introduisons la notion de séparateur, dont nous aurons besoin par la suite. Il s'agit de l'intersection de deux nœuds, c'est à dire de leurs variables en commun. Plus précisément, si deux nœuds voisins C_i et C_j ont pour domaines S_i et S_j, alors $C_i \cap C_j$ a pour domaine S_{ij}, appelé séparateur de C_i et C_j.

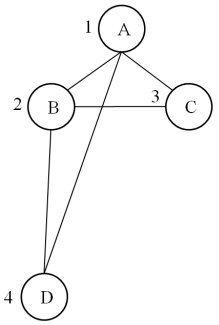
Le *junction tree* final est représenté sur la Figure 3.7.



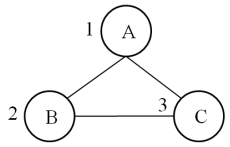
(a) Élimination de G, C_6 = {F,G}



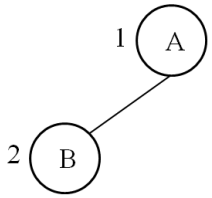
(b) Élimination de F, C_5 = {B,C,F}



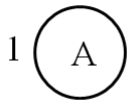
(c) Élimination de D, C_4 = {A,B,D}



(d) Élimination de C, C_3 = {A,B,C}



(e) Élimination de B, C_2 = {A,B}



(f) Élimination de A, C_1 = {A}

FIGURE 3.6 – Procédure d'élimination des variables appliquée au graphe de la Figure 3.5b

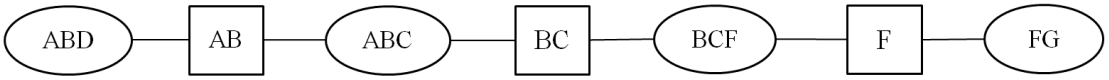


FIGURE 3.7 – *junction tree* du graphe de la Figure 3.3c

Algorithme de *Message Passing*

Nous avons expliqué que l'objectif de l'inférence est de calculer un ensemble de distributions de probabilités conjointes des variables du réseau. Le *junction tree* que nous avons construit rassemble dans chacune de ses cliques un ensemble de variables, dont nous allons maintenant calculer les distributions conjointes grâce à l'algorithme de *Message Passing*.

Il s'agit d'un processus visant à propager les informations à travers le *junction tree*, dans un sens puis dans l'autre, à la manière de l'algorithme *Forward-Backward* (cf section 5.3).

Dans le cadre de cet algorithme, on définit deux entités :

— Le **potentiel** ψ_j, proportionnel à la distribution conjointe des variables supposée par le nœud j du *junction tree*, initialisé à la valeur ψ_j^0, correspondant au produit des distributions de probabilités de chacune des variables qu'il contient. Chaque variable doit être comptée exactement une seule fois. C'est à dire que, dans notre exemple,

P(A) peut être associé indifféremment à C_3 = {A,B,C} ou C_4 = {A,B,D}, mais pas aux deux à la fois ou à aucun des deux. Afin de prendre en compte les observations, lors de l'initialisation des potentiels, toutes les composantes de ceux-ci qui sont incohérentes avec les observations sont fixées à 0, les autres étant laissées telles quelles. C'est à dire, par exemple, si A est observé à 1, $\psi_{ABD}(a, b, d)$ est fixé à 0 si $a = 0$, et laissé tel quel si $a = 1$.

— Le **Message** $\mu_{i\to j}$, un objet mathématique représentant un message de i vers j, utilisé pour permettre aux nœuds de propager leurs connaissances. Ces messages en transit entre deux nœuds sont 'stockés' dans les séparateurs, à la manière d'une mémoire tampon.

L'algorithme commence par définir un nœud quelconque du *junction tree* comme racine.

Le *Message Passing* se déroule ensuite en deux temps : une passe *Collect*, et une passe *Distribute*.

Dans la passe *Collect*, les nœuds transmettent de proche en proche leurs connaissances, depuis les feuilles jusqu'à la racine de l'arbre (post-ordre). De façon imagée, la racine "collecte" toutes les informations du réseau.

Dans la passe *Distribute*, les nœuds transmettent de proche en proche leurs connaissances, depuis la racine jusqu'aux feuilles de l'arbre (pre-ordre). De façon imagée, la racine "distribue" toutes les informations du réseau qu'elle a collecté.

Le schéma de la Figure 3.8 présente la racine choisie, ainsi que le sens de transmission des messages dans le cadre de notre exemple.

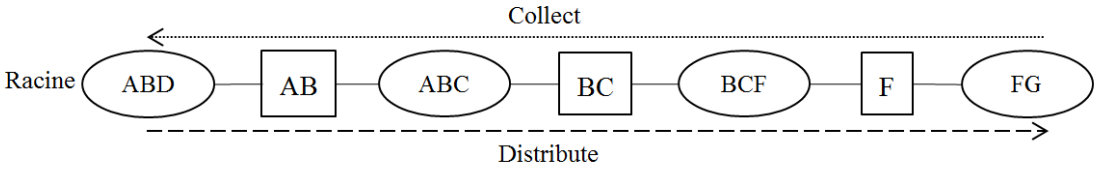


FIGURE 3.8 – Transmission des messages dans le *Message Passing* pour le *junction tree* de la Figure 3.7

Le déroulement exact de l'algorithme du *Message Passing* est le suivant :

— Passe *Collect* : pour chaque nœud j dans le post-ordre :
  — On calcule son potentiel ψ_j^c à partir des messages de chacun de ses enfants i (s'il n'y en a pas, le produit des messages vaut par défaut 1) :

$$\psi_j^c = \prod_{i \in enf(j)} [\frac{\mu_{i\to j}\psi_j^0}{\psi_j^0}] \times \psi_j^0 \tag{3.4}$$

— On transmet le message $\mu_{j\to k}$ à son unique parent k (quand il existe) :

$$\mu_{j\to k} = \sum_{S_j \setminus S_{jk}} \psi_j^c \tag{3.5}$$

$S_j \setminus S_{jk}$ signifie l'ensemble des variables de la clique j à l'exclusion des variables communes avec la clique k.

— Passe *Distribute* : pour chaque nœud j dans le pre-ordre :

— On calcule son potentiel ψ_j^d à partir du message de son unique parent k (s'il n'existe pas, le message vaut 1 par défaut) :

$$\psi_j^d = \psi_j^c \times \mu_{k\to j} \tag{3.6}$$

— On transmet les messages $\mu_{j\to i}$ à chacun de ses enfants i (quand ils existent) :

$$\mu_{j\to i} = \sum_{S_j \setminus S_{ji}} \frac{\psi_j^d}{\mu_{i\to j}} \tag{3.7}$$

Trois types d'opérations sont utilisés dans cet algorithme : la somme d'un potentiel sur des variables, le produit ou la division d'un potentiel par un message, et le produit ou la division d'un potentiel par lui-même. Nous allons expliquer en quoi consiste chacune de ces opérations.

Sommer un potentiel sur un sous-ensemble de ses variables signifie marginaliser ce potentiel sur ses autres variables. Cela se traduit, pour chaque combinaison de valeurs possible du sous ensemble de variables sur lequel on marginalise, par la sommation de toutes les composantes du potentiel possédant cette combinaison.

Cette opération est définie par la formule suivante, dans laquelle on marginalise le potentiel ψ_j ayant pour domaine S_j sur la variable X_j :

$$P(X_j) = \{P(X_j = x_j)\}_{x_j \in Domaine(X_j)},\ P(X_j = x_j) = \sum_{y_j \in Domaine(Y_j), X_j \cap Y_j = \emptyset, Domaine(X_j)+Domaine(Y_j)=S_j} P(Y_j \tag{3.8}$$

où $P(X_j = x_j)$ représente la probabilité que l'ensemble de variables X_j prenne la valeur x_j, $P(Y_j = y_j, X_j = x_j)$ est un élément de ψ_j représentant la probabilité que Y_j et X_j prennent conjointement les valeurs y_j et x_j, et Y_j représentant les variables sur lesquelles est effectué la sommation.

Multiplier (diviser) un potentiel par un message est effectué composante par composante, entre termes compatibles. On entend par là que chaque composante du potentiel est multipliée (divisée) par la composante du message dont la combinaison de valeurs des variables coïncide avec la sienne. Comme le message ne contient qu'uniquement l'information d'un sous-ensemble des variables du potentiel, la coïncidence n'est bien entendu évaluée que sur ce sous-ensemble.

Afin de définir rigoureusement cette opération, on introduit d'abord le message $\mu_{i\to j}$:

$$\mu_{i\to j} = \{P_{ij}(\bigcap_{l_{ij} \in L_{ij} \subset K_j} X_{l_{ij}} = x_{l_{ij}}^{q_{ij}})\}_{\forall q_{ij} \in Q_{ij}}. \tag{3.9}$$

$P_{ij}(z)$ représente la probabilité de l'évènement z selon $\mu_{i\to j}$. L_{ij} représente l'ensemble des variables de $\mu_{i\to j}$, et K_j l'ensemble des variables de ψ_j. $X_{l_{ij}}$ est la l_{ij}ème variable de L_{ij}, et $x_{l_{ij}}^{q_{ij}}$ est le l_{ij}ème élément de la combinaison de valeurs q_{ij} des variables de $\mu_{i\to j}$. Q_{ij} représente l'ensemble des combinaisons de valeurs des variables de $\mu_{i\to j}$.

On définit ensuite le potentiel ψ_j :

$$\begin{aligned}\psi_j &= \{P_j(\bigcap_{k_j\in K_j} X_{k_j} = x_{k_j}^{q_j})\}_{\forall q_j\in Q_j} \\ &= \{P_j(\bigcap_{l_{ij}\in L_{ij}\subset K_j} X_{l_{ij}} = x_{l_{ij}}^{q_{ij}}, \bigcap_{m_j\in M_j=K_j-L_{ij}} X_{m_j} = x_{m_j}^{q_{j\setminus ij}})\}_{\forall q_{ij}\in Q_{ij}, q_{j\setminus ij}\in Q_j\setminus Q_{ij}}.\end{aligned} \tag{3.10}$$

$P_j(z)$ représente la probabilité de l'évènement z selon ψ_j. Q_j représente l'ensemble des combinaisons de valeurs des variables de ψ_j. M_j représente l'ensemble des variables de ψ_j ne faisant pas partie des variables de $\mu_{i\to j}$, et $Q_j\setminus Q_{ij}$ est l'ensemble des combinaisons de valeurs des variables de ψ_j ne faisant pas partie des variables de $\mu_{i\to j}$.

On présente enfin le produit de $\mu_{i\to j}$ par ψ_j :

$$\begin{aligned}\mu_{i\to j}\times\psi_j =&\{P_{ij}(\bigcap_{l_{ij}\in L_{ij}\subset K_j} X_{l_{ij}} = x_{l_{ij}}^{q_{ij}}) \\ &P_j(\bigcap_{l_{ij}\in L_{ij}\subset K_j} X_{l_{ij}} = x_{l_{ij}}^{q_{ij}}, \bigcap_{m_j\in M_j=K_j-L_{ij}} X_{m_j} = x_{m_j}^{q_{j\setminus ij}})\}_{\forall q_{ij}\in Q_{ij}, q_{j\setminus ij}\in Q_j\setminus Q_{ij}}.\end{aligned} \tag{3.11}$$

Multiplier (diviser) un potentiel par lui même revient à multiplier (diviser) chacune de ses composantes par elle-même, c'est-à-dire une multiplication (division) terme à terme. Ces opérations ne sont utilisées que pour l'exactitude mathématique des produits matriciels en jeu, sans avoir à introduire de nouvelles entités.

On précise que dans le cas indéterminé $\frac{0}{0}$, le résultat est 0.

À l'issue des deux passes, les dernières valeurs de chacun des ψ_j^d sont proportionnelles à $\mathrm{P}(S_j|e)$, où S_j représente l'ensemble des variables de la clique j et e l'observation. Il suffit ensuite, pour chaque clique, de normaliser chaque élément de la distribution conjointe par rapport à l'ensemble de la distribution conjointe pour obtenir $\mathrm{P}(S_j|e)$.

La formule de normalisation est la suivante :

$$P(S_j|e) = \psi_j^{norm} = \{\psi_{j,i}^{norm}\}_{i\in I},\ \psi_{j,i}^{norm} = \frac{\psi_{j,i}^d}{\sum_{i\in I}\psi_{j,i}^d}, \tag{3.12}$$

où ψ_j^{norm} représente ψ_j^d normalisé, $\psi_{j,i}^{norm}$ représente le terme i normalisé de ψ_j^d, $\psi_{j,i}^d$ représente le terme i de ψ_j^d, et I représente l'ensemble des termes de ψ_j^d.

La procédure est présentée formalisée dans l'Algorithme 6.

Algorithme 6 : Algorithme de *Message Passing*

Données : *junction tree* J à N cliques, pre-ordre π, potentiels initiaux $\psi^0_j, j \in [1, N]$
Résultat : Ensemble des $P(S_j|e), j \in [1, N]$

pour $j = N$ *à* 1 *par pas de* -1 **faire**
  Récupérer les $\mu_{enf(\pi_j)\to\pi_j}$;
  $\psi^c_{\pi_j} = \prod_{i \in enf(\pi_j)} [\frac{\mu_{i\to\pi_j}\psi^0_{\pi_j}}{\psi^0_{\pi_j}}] \times \psi^0_{\pi_j}$;
  $\mu_{\pi_j \to par(\pi_j)} = \sum_{S_{\pi_j} \setminus S_{\pi_j par(\pi_j)}} \psi^c_{\pi_j}$;
  Envoyer $\mu_{\pi_j \to par(\pi_j)}$;
fin
pour $j = 1$ *à* N *par pas de* 1 **faire**
  Récupérer $\mu_{par(\pi_j)\to\pi_j}$;
  $\psi^d_{\pi_j} = \psi^c_{\pi_j} \times \mu_{par(\pi_j)\to\pi_j}$;
  $\mu_{\pi_j \to enf(\pi_j)} = \sum_{S_{\pi_j} \setminus S_{\pi_j enf(\pi_j)}} \frac{\psi^d_{\pi_j}}{\mu_{enf(\pi_j)\to\pi_j}}$;
  Envoyer les $\mu_{\pi_j \to enf(\pi_j)}$;
  Normaliser $\psi^d_{\pi_j}$;
fin

Si l'on souhaite extraire les distributions de certains sous-ensembles de variables de S_j, on peut ensuite marginaliser $\mathrm{P}(S_j|e)$ sur ce sous-ensemble.

Pour un exemple d'application de l'algorithme de *Message Passing* à notre réseau exemple, le lecteur pourra consulter l'Annexe 4.

3.1.3 Inférence de Viterbi dans un réseau Bayésien

Nous avons vu l'inférence classique dans les réseaux Bayésiens, consistant à trouver un ensemble de distributions de probabilités conjointes, qui permettent ensuite de calculer facilement n'importe quelle distribution de probabilité du réseau.

Nous précisons que l'algorithme de Viterbi est conçu pour être utilisé dans une chaîne de Markov (qui est un réseau Bayésien particulier), cependant, son principe sous-jacent est applicable à n'importe quel réseau Bayésien. C'est sur ce principe que nous nous appuierons dans cette section pour construire une adaptation de Viterbi aux réseaux Bayésiens quelconques.

L'inférence de Viterbi vise à trouver l'état du réseau le plus probable compte tenu des probabilités connues du réseau et des observations effectuées sur les variables. Il existe deux différences avec l'inférence classique : d'abord, le résultat est une combinaison d'états des variables du réseau (chaque variable est affectée à un état unique), avec sa probabilité associé, alors que dans l'inférence classique, on obtient un ensemble de distributions de probabilités conjointes (donc des lois). Ensuite, dans l'inférence classique, les variables sont considérées indépendamment d'une clique à l'autre, ce qui signifie que si l'on cherche l'état le plus probable

de deux variables dans deux cliques différentes, il est impossible de savoir s'ils sont compatibles, tandis qu'avec le principe de Viterbi les variables sont toutes considérées simultanément, donc on obtient l'état global le plus probable. Le fait de considérer tout le réseau et de ne pas le découper en cliques contraint cependant à ne pas considérer toutes les combinaisons d'état, mais seulement la plus probable.

L'inférence de Viterbi se base sur la propriété d'indépendance conditionnelle dans les réseaux Bayésiens. On définit dans ce cadre la notion de **frontière de Markov** d'un ensemble de variables du réseau comme l'ensemble minimal de variables du réseau nécessaire pour complètement définir les variables considérées. De manière plus formelle, dans un réseau comportant un ensemble S de variables, si l'on considère un sous-ensemble de variables Y, alors une frontière de Markov S_Y de Y est un sous-ensemble minimal de variables de S dont la connaissance rend Y indépendant du reste du réseau $S\backslash(S_Y \cup Y)$.

Cette définition se formule mathématiquement comme suit :

$$Y \perp S\backslash(S_Y \cup Y)|S_Y, \tag{3.13}$$

où X\Y signifie X exclu de Y, et X|Y signifie X sachant Y. Par sous-ensemble minimal, on entend qu'en enlever une seule variable romprait la propriété d'indépendance conditionnelle.

De façon pratique, la frontière de Markov correspond à l'ensemble des parents des variables considérées, ainsi qu'à leurs enfants et les parents de leurs enfants. Si l'on reprend l'exemple de la section précédente, comme illustré sur la Figure 3.9, la frontière de Markov du nœud B serait constituée des nœuds A (parent et parent d'enfant), D (enfant), F (enfant), et C (parent d'enfant).

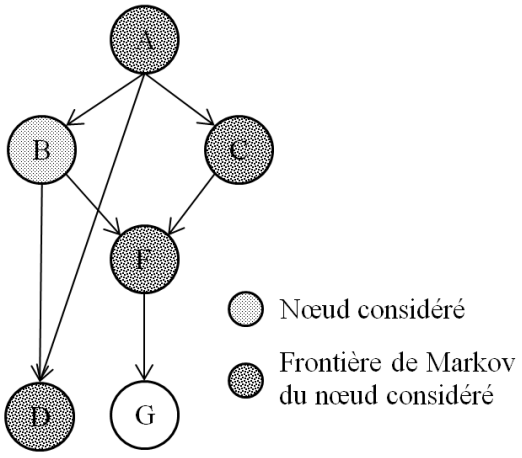


FIGURE 3.9 – Exemple de détermination de la frontière de Markov

Fait important, la définition peut être retournée, à savoir que si S_Y est une frontière de Markov de Y, alors S_Y est aussi une frontière de Markov de $S\backslash(S_Y \cup Y)$, car la propriété d'indépendance est commutative.

Maintenant que la notion de frontière de Markov est établie, nous allons l'utiliser dans l'inférence de Viterbi pour calculer l'état le plus probable du réseau. Le principe général consiste à "couper" de façon récurrente le réseau à l'aide de frontières de Markov. Étant donné un sous-réseau A d'un réseau R, le sous-réseau constitué des nœuds de R non inclus dans A peut être 'coupé' en deux parties : un sous-réseau B et une frontière de Markov (qui est aussi un sous-réseau) séparant A et B. Il est à noter que le sous-réseau B peut être vide si la frontière de Markov du sous-réseau A comprend l'intégralité des nœuds du réseau R non compris dans le sous-réseau A. La frontière de Markov peut être également vide si le sous-réseau A comprend l'intégralité des nœuds du réseau R.

Donc, si l'on considère n'importe quel sous-réseau du réseau R et le nommons A, alors le reste des nœuds du réseau R est découpable en deux sous-réseaux : B, et F la frontière de Markov entre A et B. Ensuite, si l'on considère le nouveau réseau R comme constitué de B et F, et considérons la frontière de Markov F comme le nouveau sous-réseau A, la même opération peut-être répétée, et ce jusqu'à ne plus pouvoir couper le réseau, c'est à dire lorsque B devient vide. Par exemple, appliqué à notre exemple, si l'on prend la variable A comme sous-réseau A, alors le sous-réseau F est constitué des variables B, C, et D, et le sous-réseau B des variables F et G, comme illustré sur la Figure 3.10.

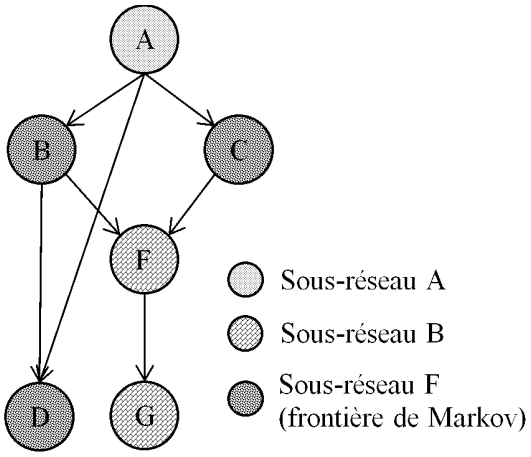


FIGURE 3.10 – Exemple de "découpage" à l'aide de la frontière de Markov

À ce stade, si l'on rassemble le sous-réseau A de départ ainsi que tous les sous-réseaux F successifs, on obtient un ensemble de frontières de Markov contenant l'intégralité des nœuds du réseau. Chacune de ces frontières, par définition, ne dépendant que de celles la jouxtant directement, il n'est donc pas nécessaire de considérer plus de deux frontières de Markov consécutives simultanément pour effectuer les calculs de probabilités. Pour prendre en compte la dépendance aux deux frontières voisines, la fenêtre est glissante, c'est à dire qu'on considère les frontières n-1 et n, puis n et n+1.

Dans le cas de l'inférence de Viterbi, on cherche à propager les informations probabilistes d'une frontière de Markov à la suivante de façon à balayer l'intégralité du réseau en maîtrisant la complexité de l'opération. Pour ce faire, la distribution conjointe du sous-réseau A de départ doit être complètement connue. De façon pratique, cela se traduit par un sous-réseau

de départ en 'amont' du réseau, c'est à dire n'ayant aucun parent.

Si l'on applique la procédure de détermination des frontières de Markov à l'exemple de la Figure 3.3c, le point de départ serait la variable A, la première frontière de Markov les variables B, D, et C, la seconde F, et la troisième G, comme illustré sur la Figure 3.11.

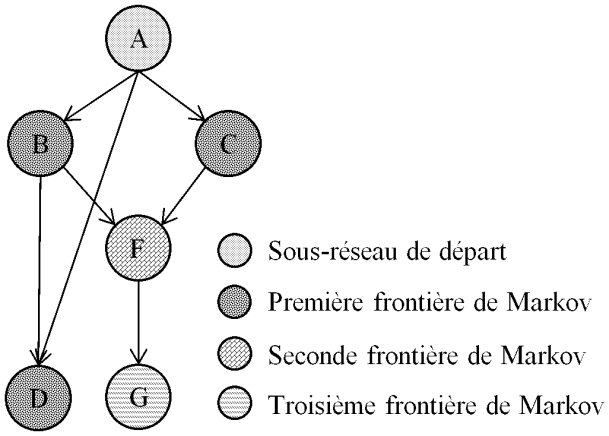


FIGURE 3.11 – Découpage d'un réseau en suite de frontières de Markov

Concrètement, l'algorithme d'inférence de Viterbi est appliqué à l'enchaînement de frontières de Markov, et se déroule en trois temps :

1. Le calcul de la probabilité de chaque état possible du sous-réseau de départ, sachant les observations (c'est à dire que si un état est incompatible avec les observations, sa probabilité est nulle).
2. Pour chaque frontière de Markov et chaque état possible de cette frontière, l'enregistrement de l'état de la frontière précédente permettant d'obtenir la probabilité la plus élevée de l'état de la frontière actuelle, et calcul de cette probabilité.
3. Récupération des états des frontières de Markov successives permettant d'obtenir la probabilité la plus élevée à l'issue du balayage du réseau. Ils constituent l'état global du réseau le plus probable.

La procédure d'inférence de Viterbi est formalisée dans l'Algorithme 7.

O_o^v désigne l'observation de la variable v à la réalisation o, $Etats(FM_n)$ désigne l'ensemble des états possibles de la frontière de Markov n, et i_v désigne la valeur de la variable v dans l'état de la frontière de Markov i. ${}^jFM_n^i$ désigne l'état de la variable j de la frontière de Markov n dans l'état i. A_n^i désigne l'élément de la structure de stockage de antécédents associé à l'état i de la frontière de Markov n. $par(j) = ({}^{par(j)}FM_{n-1}^i, {}^{par(j)}FM_{n-1}^k)$ signifie que les parents de j peuvent se situer à la fois dans la frontière de Markov courante et dans la précédente. S_n désigne le n^{ime} élément de S.

Pour terminer, il faut préciser qu'il existe une infinité de suites de frontières de Markov possibles pour balayer un réseau, et le choix d'une bonne suite est le principal levier d'optimisation de cet algorithme. En effet, si l'approche générale consiste à choisir le moins de

Algorithme 7 : Algorithme de Viterbi appliqué aux RBs

Données : réseau R, ensemble FM des N frontières de Markov du réseau, ensemble des observations O

Résultat : État le plus probable du réseau S et sa probabilité P_{max}

pour $o = 1$ *à* $Card(O)$ **faire**
  pour $i = 1$ *à* $Card(Etats(FM_1))$ **faire**
    si $\nexists v \in FM_1 / O_o^v \neq i_v$ **alors**
      $P(FM_1^i) = \prod_{j \in FM_1} P(j = {}^jFM_1^i | par(j) = {}^{par(j)}FM_1^i)$;
    sinon
      $P(FM_1^i) = 0$;
    fin
    $A_1^i = 0$;
  fin
  pour $n = 2$ *à* N *par pas de* 1 **faire**
    pour $i = 1$ *à* $Card(Etats(FM_n))$ **faire**
      si $\nexists v \in FM_n / O_o^v \neq i_v$ **alors**
        $P(FM_n^i) = \max_{k \in FM_{n-1}} P(FM_{n-1}^k)$
        $\prod_{j \in FM_n} P(j = {}^jFM_n^i | par(j) = ({}^{par(j)}FM_n^i, {}^{par(j)}FM_{n-1}^k))$;
        $A_n^i = \operatorname{argmax}_{k \in FM_{n-1}} P(FM_{n-1}^k)$
        $\prod_{j \in FM_n} P(j = {}^jFM_n^i | par(j) = ({}^{par(j)}FM_n^i, {}^{par(j)}FM_{n-1}^k))$;
      sinon
        $P(FM_n^i) = 0$;
        $A_n^i = 0$;
      fin
    fin
  fin
  $P_{max} = \max_{i \in FM_N} P(FM_N^i)$;
  $S_N = \operatorname{argmax}_{i \in FM_N} P(FM_N^i)$;
  pour $n = N$ *à* 2 *par pas de* -1 **faire**
    $S_{n-1} = A_n^{S_n}$;
  fin
fin

variables possible pour chaque tranche, il n'existe pas une unique frontière de Markov d'un ensemble de variables dès lors que l'on considère le fait qu'une variable peut être vue comme sa propre frontière de Markov : $Y \perp S \backslash Y | Y$. Cette propriété permet de conserver une variable d'une frontière de Markov à la suivante, même si cela n'est pas nécessaire, et ainsi les suites de frontières de Markov possibles deviennent infinies. Trouver la meilleure, ou au moins une bonne, est l'objet d'une étude en soit, que nous n'aborderons pas ici.

Nous avons maintenant vu comment effectuer les deux types d'inférence qui nous intéressent dans le cadre de cette thèse, nous donnant ainsi les bases nécessaires pour une autre opération fondamentale : l'apprentissage.

3.1.4 Apprentissage de paramètres dans les réseaux Bayésiens avec l'algorithme *Expectation-Maximization* (EM)

L'inférence permet de calculer n'importe quelle distribution de probabilités impliquant les variables du réseau étant donné les probabilités marginales et conditionnelles définies à la création du réseau. Mais qu'en est-il lorsque ces probabilités ne sont pas connues ? C'est là qu'entre en jeu le mécanisme d'apprentissage de paramètres, visant à approcher autant que possible les distributions caractéristiques du réseau à partir des liens existant dans le réseau et des observations disponibles. On entend par paramètres l'ensemble des éléments des tables de probabilités de chacun des nœuds du réseau.

Il existe de multiples algorithmes pour effectuer cet apprentissage, mais nous présenterons ici celui utilisé dans le cadre de cette thèse, l'un des plus couramment utilisé, l'algorithme *Expectation-Maximization* (EM).

Cette section se base sur l'explication de l'application de l'algorithme EM aux réseaux Bayésiens présentée dans [68].

Principe de l'algorithme EM

L'algorithme EM vise à trouver le jeu de paramètres du modèle d'un système maximisant la vraisemblance des observations de ce système.

Il consiste en deux étapes :

— La phase *Expectation*, visant à calculer, par inférence, l'espérance de la vraisemblance du système sachant la valeur du jeu de paramètres actuel et les observations. Dans sa version logarithmique, cette espérance de vraisemblance (aussi appelée espérance de log-vraisemblance) s'écrit : $Q(\Theta, \Theta^r) = \mathbb{E}(L(X;\Theta)|^o X, \Theta^r)$, avec $\mathbb{E}$ représentant l'opérateur espérance, X l'ensemble les variables du modèle, toutes réalisations confondues, Θ le jeu de paramètres de X, ${}^oX$ l'ensemble des observations du système, toutes réalisations confondues, et Θ^r, la valeur actuelle du jeu de paramètres Θ.

— La phase *Maximization*, visant à trouver une nouvelle valeur du jeu de paramètres du modèle, maximisant l'espérance de la vraisemblance des observations, c'est à dire en maximisant $Q(\Theta, \Theta^r)$. Cela s'écrit $\Theta^{r+1} = \underset{\Theta}{\operatorname{argmax}}\, Q(\Theta, \Theta^r)$

Les paramètres sont judicieusement (ou aléatoirement) initialisés au début de l'algorithme, puis les phases *Expectation* et *Maximization* sont répétées alternativement jusqu'à convergence vers un maximum local (qu'on espère global).

L'algorithme EM appliqué aux réseaux Bayésiens

Appliqué aux réseaux Bayésiens, $Q(\Theta, \Theta^r)$ s'exprime comme suit :

$$Q(\Theta, \Theta^r) = \sum_{i,j_i,k_i} log(\theta_{i,j_i,k_i}) \widehat{N_{i,j_i,k_i}}, \tag{3.14}$$

où $\theta_{i,j_i,k_i} = P(X_i = k_i | par(X_i) = j_i)$ représente un paramètre du réseau sachant ses parents, $\widehat{N_{i,j_i,k_i}} = \sum_{m=1}^{M} P(par(X_i^m) = j_i, X_i^m = k_i | {}^o X^m, \Theta^r)$, et m représente une réalisation du réseau (qui se matérialise par les observations effectuées lors de cette réalisation).

La phase *Expectation* consiste donc à calculer $\widehat{N_{i,j_i,k_i}}$ par inférence, puis à en déduire $Q(\Theta, \Theta^r)$. La phase *Maximization* consiste à maximiser $\sum_{k_i} log(\theta_{i,j_i,k_i}) \widehat{N_{i,j_i,k_i}}$ sous les contraintes $\sum_{k_i} \theta_{i,j_i,k_i} = 1$ et $0 \leq \theta_{i,j_i,k_i} \leq 1$, pour toutes valeurs du triplet (i, j_i, k_i).

À l'aide de la méthode des multiplicateurs de Lagrange, on obtient l'équation suivante de mise à jour du jeu de paramètres Θ :

$$\Theta_{i,j_i,k_i}^{r+1} = \frac{\widehat{N_{i,j_i,k_i}}}{\sum_{k_i} \widehat{N_{i,j_i,k_i}}}. \tag{3.15}$$

Pour de plus amples détails sur l'obtention des formules de cette section, le lecteur pourra consulter l'Annexe 5.

3.2 Réseaux Bayésiens dynamiques

Pour la présentation des réseaux Bayésiens dynamiques, nous supposons que le lecteur est déjà relativement familier avec les chaînes de Markov. Si ce n'est pas le cas, le lecteur peut les revoir en Annexe 2.

3.2.1 Principes fondamentaux et représentation des réseaux Bayésiens dynamiques

Les réseaux Bayésiens dynamiques (RBD) sont des réseaux Bayésiens particuliers pouvant être vus comme une généralisation des Chaînes de Markov (CM). En effet, ils combinent un nombre illimité de variables pouvant être liées conditionnellement sans autre restriction que l'absence de cyclicité (RB) à la notion de temporalité, et plus particulièrement d'invariance du comportement dans le temps (CM). On peut donc voir un réseau Bayésien dynamique comme une chaîne de Markov dotée d'un nombre quelconque de variables cachées et observables.

D'un point de vue pratique, cela se traduit par le fait que le réseau Bayésien est formé d'une répétition du même motif un nombre déterminé de fois, chaque motif représentant l'état du système à un instant donné. Ces motifs, appelés **tranches temporelles**, sont liés entre eux par des liens conditionnels comme au sein des tranches. Il découle de cela que les notions de variable du système et variable du réseau ne sont plus équivalentes : en effet, une variable du réseau est en réalité une variable du système à un instant donné. De plus, le comportement d'une variable du système étant invariant dans le temps, les différentes variables du réseau

relevant de la même variable du système partagent une unique table de probabilité.

Bien que n'importe quelle relation entre les tranches temporelles est théoriquement possible (tant que l'on ne crée pas de cycles), et exacte d'un point de vue mathématique, généralement cela n'a pas de sens d'un point de vue physique, et donc on utilisera la plupart du temps (et dans cette thèse) uniquement des dépendances conditionnelles d'un instant aux instants précédents.

Le principal avantage de ce type de réseau est de conserver la complexité des relations d'un réseau Bayésien tout en obtenant la possibilité des chaînes de Markov de générer un réseau s'étendant sur un nombre infini d'instants, mais possédant un nombre fini de paramètres (donc avec la possibilité d'effectuer de l'apprentissage).

La Figure 3.12 illustre la représentation classique d'un RBD quelconque d'ordre 1, comportant N variables par tranche. Cette représentation comporte certains problèmes de lourdeur lorsque l'ordre (à prendre au même sens que dans les chaînes de Markov) augmente, et que l'on doit donc représenter n+1 tranches temporelles pour un ordre n.

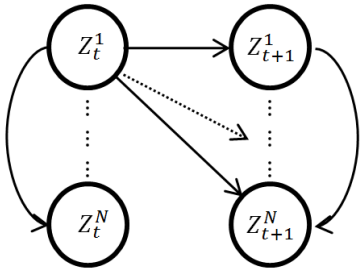


FIGURE 3.12 – Représentation classique, dite "2 tranches" d'un RBD générique

3.2.2 Inférence exacte dans un réseau Bayésien dynamique

La procédure d'inférence dans un RBD s'appuie sur celle d'inférence dans un RB, utilisant l'algorithme de *Message Passing* en sous-routine d'un algorithme plus général, l'algorithme d'interface. Cette méthode d'inférence n'est bien évidemment pas la seule, mais elle a le mérite d'être simple, efficiente, et exacte. Il s'agit de l'une des contributions de la thèse de K. P. Murphy [67]. À noter que dans cette dernière, seul le cas d'ordre un est présenté, mais nous généraliserons ici à un ordre N quelconque, d'où un changement de certaines notations et définitions par rapport au document d'origine. De plus, nous définirons de nouvelles entités pour combler des manques d'explication de certaines étapes clés.

La démarche générale reste la même, mais avec plus d'étapes, que nous allons détailler avec exemple à l'appui. Le schéma 3.13 donne une vue d'ensemble de la procédure complète d'inférence. Celle-ci se déroule en six étapes :

1. Moralisation des graphes
2. Triangularisation des graphes

3. Procédure d'élimination des variables
4. Assemblage de chacun des *junction trees*
5. Connexion des *junction trees*
6. Exécution de l'algorithme d'interface

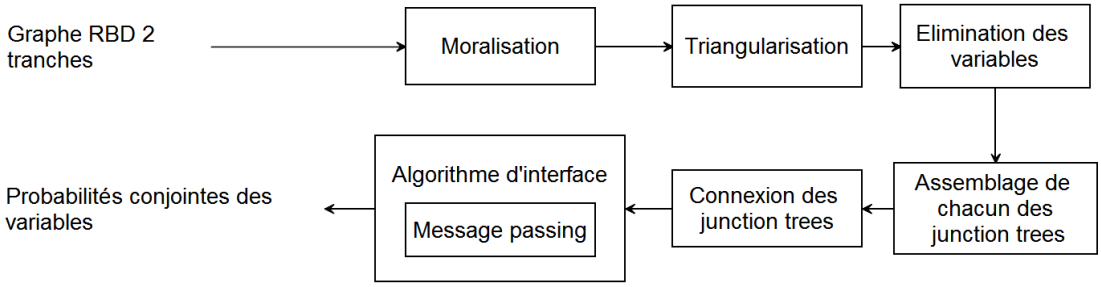


FIGURE 3.13 – Procédure d'inférence exacte dans un réseau Bayésien dynamique avec l'algorithme d'interface

Construction des *junction trees*

Les cinq premières étapes ont pour but de transformer le graphe du réseau en *junction tree*.

Tout d'abord, on définit deux notions : celle d'interface *Forward*, et celle d'interface *Backward*. L'interface *Forward* I_t^{t+n} est l'ensemble des nœuds de la tranche t ayant un arc sortant vers un nœud de la tranche $t+n$. À noter qu'une interface *Forward* pour un n donné peut être vide. L'interface *Backward* I_t^{t-n} est l'ensemble des nœuds de la tranche t ayant un arc entrant depuis un nœud de la tranche $t-n$. À noter qu'une interface *Backward* pour un n donné peut être vide.

Le schéma 3.14 met en évidence en pointillé les nœuds appartenant à I_t^{t-1} et en hachuré ceux appartenant à I_t^{t+1} sur un exemple d'ordre un, dérivant de celui employé dans la section 3.1. Trois tranches temporelles sont représentées afin de simplifier la lecture, mais l'on n'en considérera plus qu'une seule pour la suite de la construction du *junction tree*. Comme on peut le voir, un nœud peut être dans les deux interfaces à la fois.

Moralisation et triangularisation : Pour chacun des *junction trees* J_t à construire à partir d'une tranche temporelle du réseau, et pour chacune de ses interfaces I_t^{t-1} à I_t^{t-N} et I_t^{t+1} à I_t^{t+N}, il doit exister au moins une clique rassemblant tous ses nœuds.

On s'en assure en rendant le sous-graphe de chaque interface fortement connexe lors de la moralisation, c'est à dire en ajoutant des arcs bidirectionels entre tous les nœuds de chacune des interfaces. Dans le cas de notre exemple, les nœuds de l'interface *Forward*, A et F, sont donc reliés, et ceux de l'interface *Backward* étant déjà liés, aucun autre changement n'est nécessaire. Après moralisation et triangularisation, on obtient le graphe de la Figure 3.15

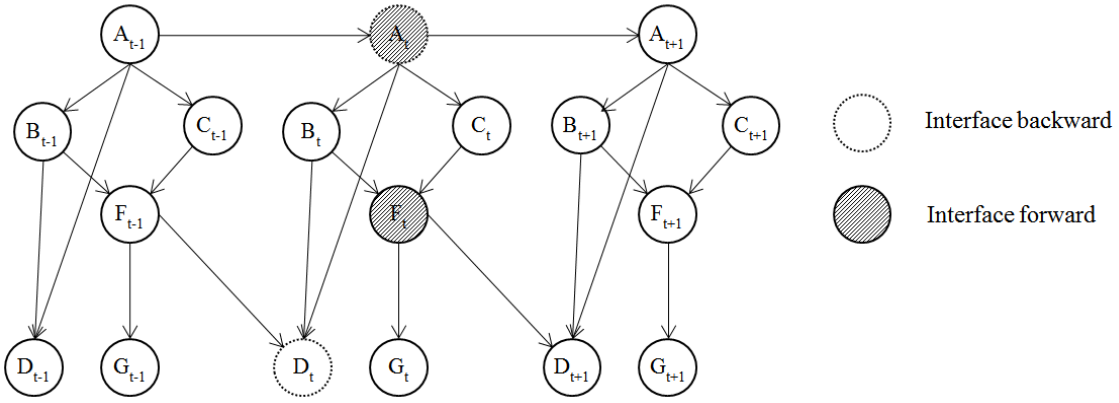


FIGURE 3.14 – Mise en évidence des interfaces *Forward* I_t^{t+1} et *Backward* I_t^{t-1} sur un exemple

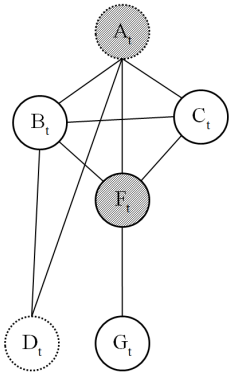


FIGURE 3.15 – Représentation "1 tranche" du graphe de la Figure 3.14 moralisé et triangularisé

Élimination des variables et assemblage des *junction trees* : On applique ensuite les procédures d'élimination des variables et de construction des *junction trees* comme décrit dans la section 3.1. Dans le cas de notre exemple, et avec l'ordre d'élimination $\pi = \{A_t, B_t, C_t, D_t, F_t, G_t\}$, on obtient, après suppression des doublons, les cliques suivantes : $C_t^4 = \{A_t, B_t, D_t\}, C_t^5 = \{A_t, B_t, C_t, F_t\}, C_t^6 = \{F_t, G_t\}$. Le *junction tree* obtenu pour une tranche t quelconque du réseau est présenté sur la Figure 3.16.

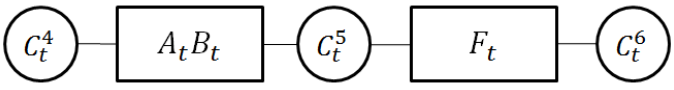


FIGURE 3.16 – *junction tree* de la tranche temporelle t du réseau exemple

Connexion des *junction trees* : L'étape suivante consiste à connecter tous les *junction trees* J_1 à J_T entre eux pour permettre la propagation de l'information d'un instant à l'autre. Cette connexion est réalisée via les interfaces *Forward* ; I_{t-n}^t pour les connexions entre J_{t-n} et J_t et I_t^{t+n} pour celles entre J_t et J_{t+n}, et ce pour tout t compris entre 1 et T et tout n compris entre 1 et N. On appellera le nouveau *junction tree* ainsi formé J.

Le schéma de la Figure 3.17 illustre le principe général de ces connexions du point de vue d'un *junction tree* J_t quelconque non terminal ou initial. D_t^N représente une clique de J_t contenant l'interface I_t^{t-N}, et C_t^N une clique de J_t contenant I_t^{t+N}, car n'importe quelle clique de J_t contenant l'interface voulue peut-être utilisée. Les rectangles entre les *junction tree*s représentent les séparateurs des junctions trees, à savoir leurs interfaces *Forward*. Enfin, N_t représente toutes les autres cliques du *junction tree* J_t.

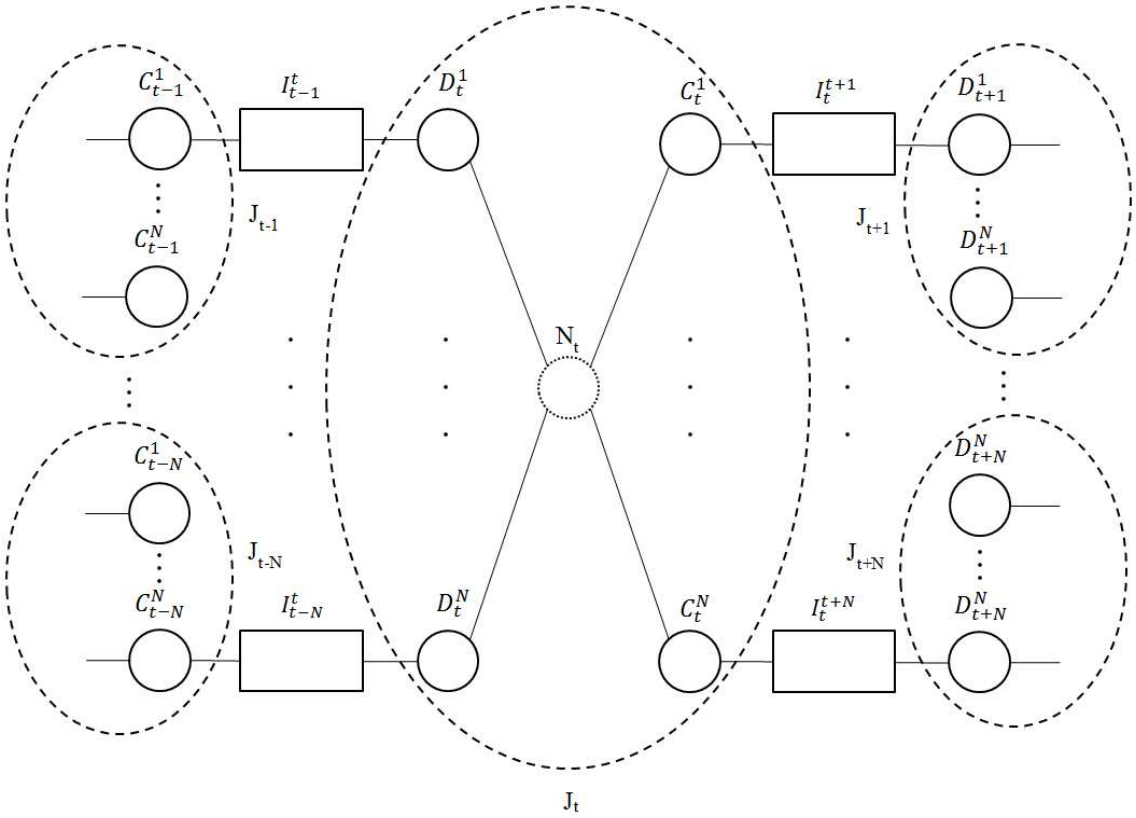


FIGURE 3.17 – Principe de connexion des *junction tree*s pour un réseau d'ordre N

Dans le cas de notre exemple, chaque *junction tree* J_t n'est lié qu'à ses deux voisins, à savoir à J_{t-1} via I_{t-1}^t et à J_{t+1} via I_t^{t+1}, le principe de connexion est donc simplifié, comme illustré sur la Figure 3.18.

Le schéma de la Figure 3.19 présente un *junction tree* J_t quelconque non terminal de la chaîne de *junction tree*s de notre exemple, et ses connexions avec les *junction tree*s voisins J_{t-1} et J_{t+1}. Les variables A_{t-1} et F_{t-1} ne sont pas explicitement incluses dans C_t^4, mais au vu des liens conditionnels entre les tranches, elles apparaissent naturellement lorsque les distributions de A_t et D_t sont considérées à un instant $t > 1$.

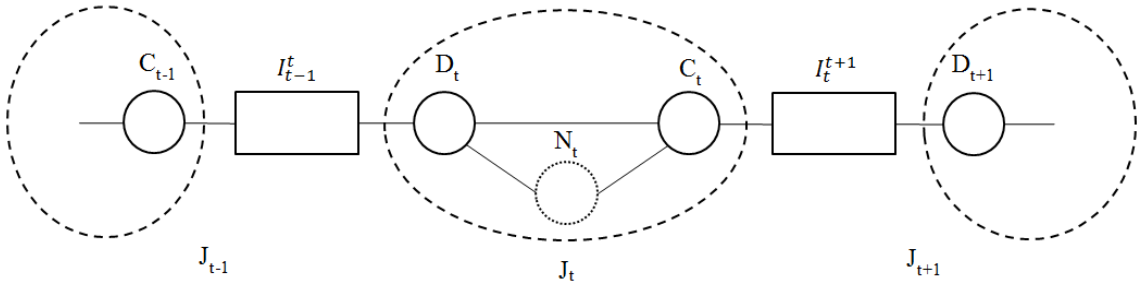


FIGURE 3.18 – Principe de connexion des *junction trees* pour un réseau d'ordre un

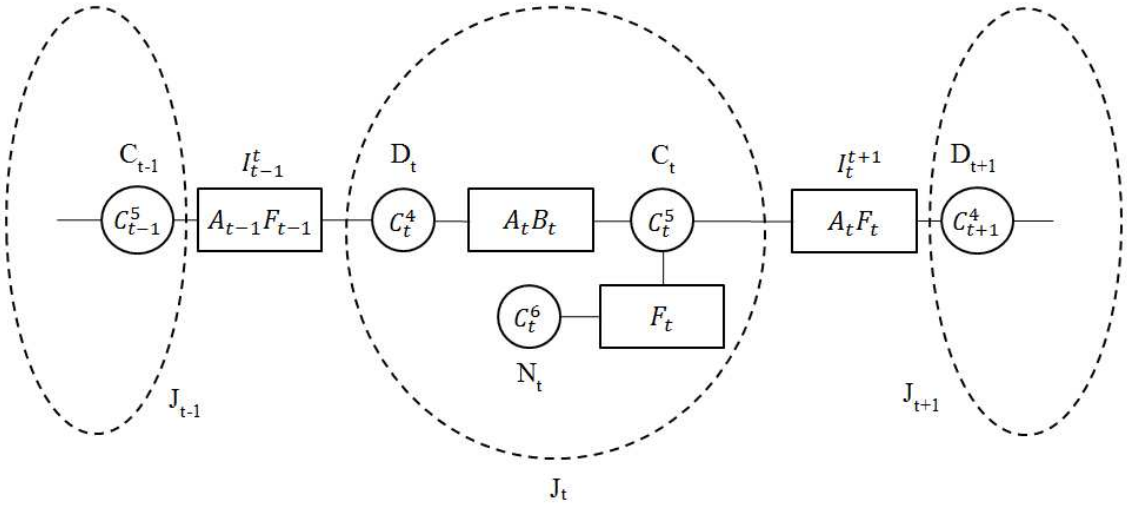


FIGURE 3.19 – Un *junction tree* J_t non terminal de notre exemple et ses connexions à ses voisins

Algorithme d'interface

Cet algorithme reprend le même principe que le *Message Passing*, mais pour des *junction trees* composites. Il propage l'information d'un instant à l'autre, dans un sens puis dans l'autre, comme le *Forward-Backward*. De plus, l'inférence au sein de chaque *junction tree* J_t est faite à l'aide de l'algorithme de *Message Passing*. Comme pour ce dernier, tous les produits et divisions sont termes à termes.

On initialise au départ les potentiels des cliques de tous les *junction trees* J_1 à J_T de la même façon que pour l'algorithme de *Message Passing*.

Ensuite, pour chaque J_t de 1 à T on effectue la passe *Forward* comme suit :

1. On marginalise les potentiels des C_{t-n}^{n} sur I_{t-n}^{t}, pour tout n entre 1 et N (quand ils existent). Cela représente l'ensemble des potentiels des interfaces *Forward* des J_{t-n} vers J_t sachant les observations effectuées jusqu'à l'instant précédent, $\{\psi(I_{t-n}^{t}|y_{1:t-1}), 1 \leq n \leq N\}$.
2. On met à jour les potentiels des D_t^n en les multipliant par l'élément d'ordre n correspondant de $\{\psi(I_{t-n}^{t}|y_{1:t-1}), 1 \leq n \leq N\}$, pour tout n entre 1 et N (quand ils existent).

3. On effectue une passe *Collect* puis *Distribute* à travers J_t avec un nœud quelconque de J_t comme racine. Si $n = 1$, une simple passe *Collect* avec le nœud C_t comme racine suffit.
4. On sauvegarde tous les potentiels des cliques de J_t et les potentiels des séparateurs entre les J_{t-n} et J_t (quand ils existent). On sauvegarde également ceux des séparateurs à l'intérieur de J_t si seulement la passe *Collect* a été effectuée.

Pour l'instant initial t = 1, on saute les étapes 1 et 2 de la passe *Forward*.

Pour chaque J_t de T à 1, on effectue la passe *Backward* comme suit :

1. On marginalise les potentiels des D^n_{t+n} sur I^{t+n}_t pour tout n entre 1 et N (quand ils existent). Cela représente l'ensemble des potentiels des interfaces *Forward* des J_{t+n} vers J_t sachant toutes les observations,$\{\psi(I^{t+n}_t|y_{1:T}), 1 \leq n \leq N\}$.
2. Pour tout n entre 1 et N, on met à jour les potentiels des C^n_t (quand ils existent) de la façon suivante :
$$\psi^b_{C^n_t} = \psi^f_{C^n_t} \times \frac{\psi(I^{t+n}_t|y_{1:T})}{\psi(I^{t+n}_t|y_{1:t-1})}, \tag{3.16}$$
où f symbolise la passe *Forward*, et b la passe *Backward*.
3. On effectue une passe *Collect* puis *Distribute* à travers J_t avec un nœud quelconque de J_t comme racine. Si $n = 1$, une simple passe *Distribute* avec le nœud C_t comme racine suffit.
4. On sauvegarde tous les potentiels des cliques de J_t, qui correspondent aux probabilités conjointes des nœuds de chaque clique de J_t, $P(S_{t,j}|e), j \in [1, M]$. M représente le nombre de cliques dans le *junction tree* J_t.

Pour l'instant final t = T, on saute les étapes 1 et 2 de la passe *Backward*.

On peut ensuite extraire des probabilités conjointes toutes les distributions de probabilités souhaitées par marginalisation, comme à l'issue de l'algorithme de *Message Passing* seul. Il est à noter que l'on doit utiliser une simple passe *Distribute* lors de la passe *Backward* si et seulement si l'on a opté pour une simple passe *Collect* lors de la passe *Forward*.

Cette procédure est formalisée dans l'Algorithme 8 lorsque n est considéré quelconque.

Ψ^f_t représente l'ensemble des potentiels *Forward* du *junction tree* J_t.

Pour une application numérique de l'algorithme d'interface sur notre exemple, voir Annexe 6.

3.2.3 Inférence de Viterbi dans un réseau Bayésien dynamique

Lorsque l'on déroule un réseau Bayésien dynamique, on obtient un simple réseau Bayésien présentant un motif qui se répète, la procédure d'inférence du Viterbi présentée dans la section 3.1.3 reste donc parfaitement applicable.

La principale différence entre l'inférence de Viterbi dans un RB et celle dans un RBD est liée à cette répétition d'un même motif, la tranche temporelle, qui donne ainsi une suite de

Algorithme 8 : Algorithme d'interface

Données : *junction tree* composite J à T *junction tree*s élémentaires, pre-ordre π, potentiels initiaux $\psi^0_{t,j}, t \in [1,T], j \in [1,M]$

Résultat : Ensemble des $P(S_{t,j}|e), t \in [1,T], j \in [1,M]$

pour $t = 1$ *à* T *par pas de* 1 **faire**
  si $t > 1$ **alors**
    pour $n = 1$ *à* N *par pas de* 1 **faire**
      $\psi(I^t_{t-n}|y_{1:t-1}) = \sum_{C^n_{t-n} \setminus I^t_{t-n}} C^n_{t-n}$;
      $\psi^0_{D^n_t} = \psi^0_{D^n_t} \times \psi(I^t_{t-n}|y_{1:t-1})$;
    fin
  fin
  $\Psi^f_t = Message\ Passing(J_t, \pi, \Psi^0_t)$;
fin
pour $t = T$ *à* 1 *par pas de* -1 **faire**
  si $t < T$ **alors**
    pour $n = 1$ *à* N *par pas de* 1 **faire**
      $\psi(I^{t+n}_t|y_{1:T}) = \sum_{D^n_{t+n} \setminus I^{t+n}_t} D^n_{t+n}$;
      $\psi^b_{C^n_t} = \psi^f_{C^n_t} \times \frac{\psi(I^{t+n}_t|y_{1:T})}{\psi(I^{t+n}_t|y_{1:t-1})}$;
    fin
  fin
  $\Psi^b_t = Message\ Passing(J_t, \pi, \Psi^b_t)$;
fin

couvertures (c'est à dire qu'il est potentiellement possible d'en retirer des nœuds sans perdre la propriété d'indépendance entre les deux parties du réseau de chaque côté) de Markov évidente, toutes identiques. Celles-ci sont composées d'une tranche temporelle R_t à laquelle sont ajoutées les variables des interfaces *Forward* des $N-1$ tranches précédentes vers les $N-1$ tranches suivantes, où N est l'ordre du réseau. On rappelle que toutes les interfaces *Forward* d'une tranche vers une autre n'existent pas forcément.

On peut formaliser cette définition de la suite des couvertures de Markov comme suit :

$$CM = \{R_t \bigcup_{n=t-N+1, n\geq 0}^{t-1} \bigcup_{p=t+1, p\leq T}^{t+N-1} I^p_n\}_{\forall t \in [1,T]}, \tag{3.17}$$

où T représente le nombre de tranches temporelles du réseau.

Sur la Figure 3.20, une couverture de Markov générique déterminée comme précédemment expliqué est présentée sur un exemple de réseau inspiré de celui vu dans la section précédente, auquel on a rajouté un lien au second ordre de F vers D. On y voit clairement qu'en prenant la tranche t ainsi que le nœud F de la tranche $t-1$, unique nœud de l'interface de $t-1$ vers $t+1$, le réseau se trouve coupé en deux parties indépendantes l'une de l'autre.

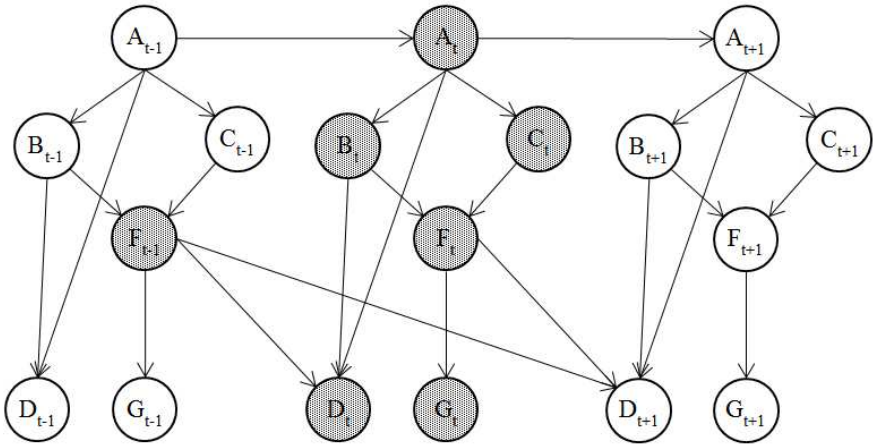


FIGURE 3.20 – Exemple de couverture de Markov évidente d'un réseau Bayésien dynamique

Bien évidement, comme pour les réseaux Bayésiens classiques, il existe une infinité d'autres couvertures de Markov possibles, certaines potentiellement meilleures, dont notamment des frontières de Markov.

3.2.4 Apprentissage dans les réseaux Bayésiens dynamiques avec l'algorithme *Expectation-Maximization* (EM)

Cette section est basée sur l'explication de l'application de l'algorithme EM aux réseaux Bayésiens présentée dans [68].

Différences avec l'apprentissage dans les réseaux Bayésiens

Dans le cas des réseaux Bayésiens classiques, on considère toutes les variables du réseau indépendamment, c'est à dire ayant leur propre loi de probabilité, et donc on y accède par leur nom uniquement, ce qui signifie que l'ensemble des variables du réseau est vu comme un vecteur.

Quant aux réseaux Bayésiens dynamiques, on considère que le réseau est constitué de variables se répétant temporellement, donc partageant la même loi de probabilité d'un instant à l'autre. Il découle de cela que l'on accède aux différentes variables par leur nom et tranche temporelle, ce qui signifie que l'ensemble des variables du réseau est vu comme une matrice.

La présence de dépendance temporelle implique par définition la nécessité d'initialisation au(x) premier(s) instant(s), et dans le cas des réseaux Bayésiens dynamiques, cela signifie qu'un nœud ayant p dépendances avec m ordres différents aura $m + 1$ **tranches de définition**. Chacune de celles-ci est liée à une table de probabilité différente, possédant ses propres paramètres, et n'existe que sur un intervalle bien précis. On précise que lorsque $m = 0$, il s'agit d'un simple lien non temporel comme ceux des réseaux Bayésiens classiques.

Pour se faire une idée des intervalles de validité des différentes tranches de définition d'un nœud, il faut prendre tous les m ordres de dépendance différents de ce nœud et les placer

sur un axe gradué de 1 à T. Chaque intervalle (borne inférieure non comprise et supérieure comprise) sur l'axe représente le domaine d'existence d'une tranche de définition.

La Figure 3.21 met en évidence les différentes tranches de définition d'un exemple de réseau d'ordre 3. On considère le nœud A, ayant pour parents à l'ordre 1 les nœuds A et B et pour parent à l'ordre 3 le nœud C. Cela signifie qu'à la tranche temporelle 1, le nœud A n'a aucun parents, aux tranches 2 et 3 il a pour parents A et B, et à partir de la tranche 4, il a pour parents A, B, et C.

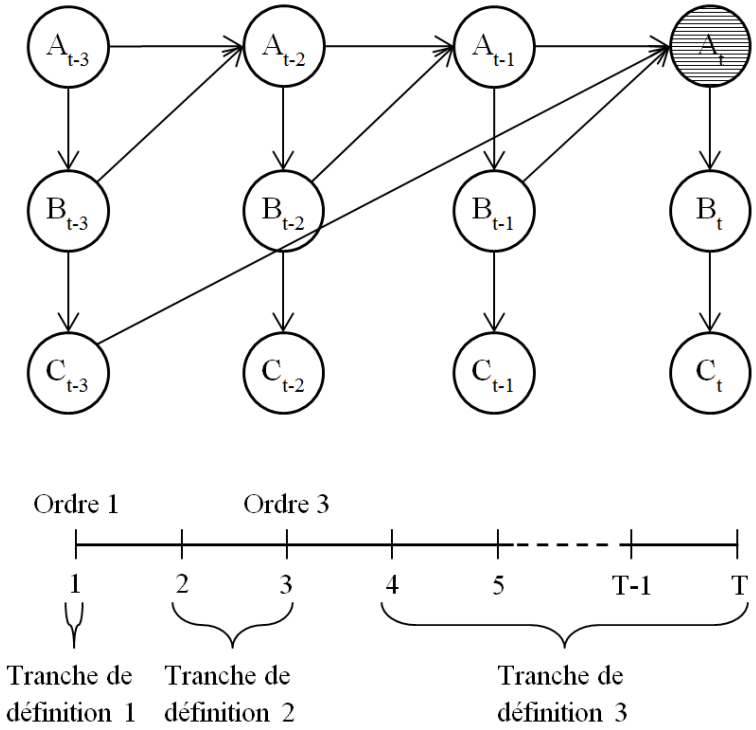


FIGURE 3.21 – Mise en évidence des tranches de définition sur un exemple

En dehors de ces quelques différences, le principe reste le même que dans la section 3.1.3.

L'algorithme EM appliqué aux réseaux Bayésiens dynamiques

Appliqué aux réseaux Bayésiens dynamiques, $Q(\Theta, \Theta^r)$ s'exprime comme suit :

$$Q(\Theta, \Theta^r) = \sum_{i,l_i,j_{l_i},k_i} log(\theta_{i,l_i,j_{l_i},k_i}) \widehat{N_{i,l_i,j_{l_i},k_i}}, \tag{3.18}$$

où $\theta_{i,l_i,j_{l_i},k_i} = P(X_i = k_i | \{par(X_i)/ordre(par(X_i)) \leq ordres\{par(X_i)\}_{l_i}\} = j_{l_i})$ représente un paramètre du réseau sachant ses parents, avec l_i la tranche de définition du nœud X_i, $ordre(par(X_i))$ l'ordre d'un parent, et $ordres\{par(X_i)\} = \{ordre(par(X_i))/\forall p, q \in |ordres\{par(X_i)\}|/p < q, ordre(par(X_i))_p < ordre(par(X_i))_q\}$ l'ensemble des ordres uniques des parents classés en ordre croissant.

$$\widehat{N_{i,l_i,j_{l_i},k_i}} = \sum_{m=1}^{M} \sum_{t=Ordres\{par(X_i)\}_{l_i}^{m}}^{Ordres\{par(X_i)\}_{l_i+1}^{m}-1} P(par(X_i^t)_{l_i} = j_{l_i,m}\, X_i^t = k_i|^o X^m, \Theta^r) \tag{3.19}$$

,avec $Ordres\{par(X_i)\}^m = ordres\{par(X_i)\} + \{0\} + \{T_m\}$, l'ensemble des ordres uniques des parents auquel on ajoute 0 et T_m afin de s'assurer que l'excursion de l'ensemble correspond bien aux tranches temporelles existant à chaque réalisation m.

On a également $par(X_i^t)_{l_i} = \{par(X_i^t)/ordre(par(X_i^t)) \leq Ordres\{par(X_i)\}_{l_i}\}$, la combinaison de parents du nœud X_i^t à la tranche de définition l_i.

La phase *Expectation* consiste donc à calculer $\widehat{N_{i,l_i,j_{l_i},k_i}}$ par inférence, puis à en déduire $Q(\Theta, \Theta^r)$. La phase *Maximization* consiste à maximiser $\sum_{k_i} log(\theta_{i,l_i,j_{l_i},k_i})\widehat{N_{i,l_i,j_{l_i},k_i}}$ sous les contraintes $\sum_{k_i} \theta_{i,l_i,j_{l_i},k_i} = 1$ et $0 \leq \theta_{i,l_i,j_{l_i},k_i} \leq 1$, pour toutes valeurs du quadruplet (i, l_i, j_{l_i}, k_i).

À l'aide de la méthode des multiplicateurs de Lagrange, on obtient l'équation suivante de mise à jour du jeu de paramètres Θ :

$$\Theta_{i,l_i,j_{l_i},k_i}^{r+1} = \frac{\widehat{N_{i,l_i,j_{l_i},k_i}}}{\sum_{k_i} \widehat{N_{i,l_i,j_{l_i},k_i}}}. \tag{3.20}$$

Pour de plus amples détails sur l'obtention des formules de cette section, le lecteur pourra consulter l'Annexe 7.

3.3 Modèle proposé : BaNet3F

Maintenant que nous avons vu les bases des réseaux Bayésiens classiques et dynamiques, nous allons pouvoir exposer le modèle proposé dans cette thèse, Bayesian Network Frame Format Finder, ou BaNet3F. Cette présentation se déroulera en cinq temps : d'abord, nous donnerons une vue générale du modèle, puis nous présenterons les quatre éléments principaux le constituant, à savoir le réseau utilisé, l'application de l'EM pour effectuer l'apprentissage des paramètres, l'inférence de Viterbi pour trouver le découpage le plus probable des trames, et la validation du découpage sur la base de sa vraisemblance.

3.3.1 Principe fondamental et vue générale du modèle

Principe fondamental

On rappelle que la problématique de cette thèse consiste à identifier la structure des trames d'une trace constituée d'un ensemble de trames contenant un protocole binaire unique.

Pour cela, un modèle basé sur un réseau Bayésien à mi-chemin entre version classique et dynamique a été conçu. Son objectif est de rassembler les bits connexes en séquences, la position de ces dernières correspondant chacune à un slot de champ.

Afin de déterminer si un bit B_t à la position t appartient au même champ que le(s) précédent(s) bit(s), le réseau compare les différentes probabilités de la valeur observée du bit B_t sachant les n bits précédents, n variant entre 0 et 7, et choisit la plus forte. L'ensemble des probabilité conditionnelles comparées est donc le suivant : $\{P(B_t|\bigcap_{i=t-n}^{t-1} B_i)\}_{n\in[0;7]}$

Le bit B_t ainsi que les n précédents sélectionnés constituent une séquence. Bien entendu, ce qui est exactement recherché, c'est l'enchaînement de séquences le plus probable ; il faut donc considérer la trame dans son ensemble pour déterminer le découpage en séquences le plus vraisemblable.

Vue générale du modèle

Afin de limiter la complexité du modèle, nous avons décidé d'opter pour une approche récursive de l'analyse de la trace. Ainsi, à chaque passe, le modèle cherche un découpage optimal des trames de la trace en un nombre maximal de fragments (paramétrable), puis évalue si ce découpage est acceptable. Il poursuit la récursion tant que le découpage est considéré possible.

Pour trouver le découpage optimal, le modèle construit d'abord le réseau sur la base des caractéristiques de la trace à analyser, puis apprend les paramètres du réseau à l'aide de l'EM. Ensuite il détermine la séquence d'états du réseau la plus probable à l'aide d'une variante de l'algorithme de Viterbi, Automatic Markov Boundaries construction optimized Viterbi (AMBV), développée dans le cadre cette thèse, de laquelle il déduit le découpage le plus probable.

Afin de réduire encore plus la complexité, on tire parti du fait qu'un champ est forcément un multiple d'octets ou bien contenu à l'intérieur d'un octet, en considérant l'octet comme unité élémentaire de découpage tant que la longueur des trames à analyser est strictement supérieure à un octet. Lorsque les trames sont découpées jusqu'à arriver à des longueurs d'un octet, alors on utilise le bit comme unité élémentaire.

Le schéma de la Figure 3.22 présente le fonctionnement global du modèle. On y trouve le dataset initial, qui n'est autre que la trace à analyser mise en forme pour pouvoir être traitée par le réseau. Dans un premier temps, en fonction de la taille du dataset à analyser, on opte pour un découpage en octets ou en bits. Ensuite, avant d'effectuer l'analyse en elle-même pour trouver le découpage optimal, on fait une analyse statistique du dataset qui servira à l'initialisation du réseau. Cette analyse a pour résultat un découpage proposé en bit ou octet, qui est ensuite soumis à la procédure de validation. Si le découpage est validé, alors pour chaque champ trouvé, on génère un nouveau dataset dans lequel on met tous les exemplaires du champ concerné à travers toutes les trames, et on ajoute ce dataset en queue de la file de ceux à analyser. Si le découpage est invalidé, alors on considère qu'il n'est plus possible de découper plus avant le dataset de départ de la passe, et on l'ajoute à la liste des champs finaux sans tenir compte du découpage proposé. Ensuite, tant qu'il reste des datasets à analyser, le premier de la queue est récupéré et toute la procédure précédente est répétée. Lorsque la file des datasets à analyser finit par être vide, on utilise la liste des champs finaux pour reconstituer la trace d'origine avec le découpage final trouvé.

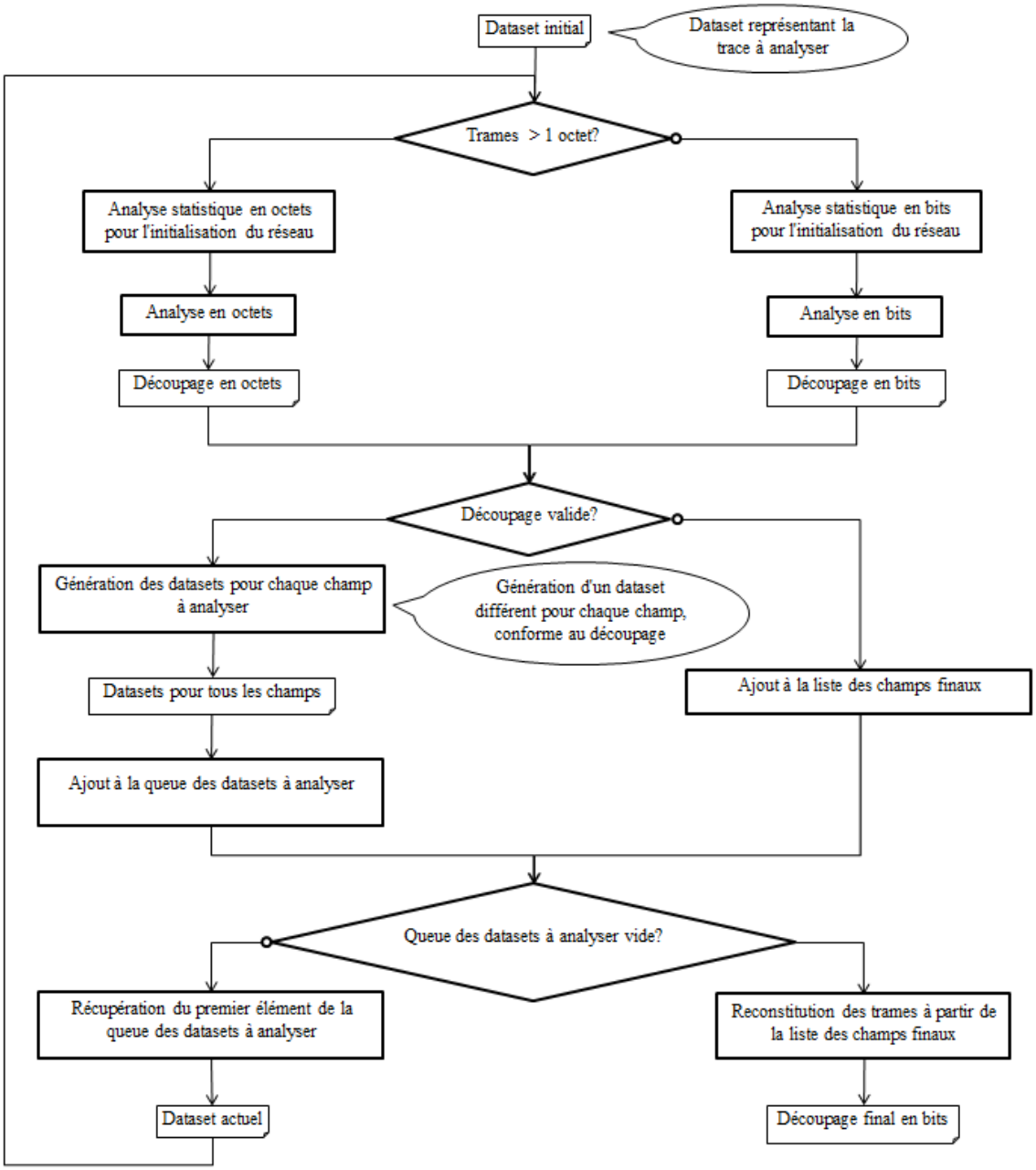


FIGURE 3.22 – Vue globale de BaNet3F

L'analyse à l'aide du réseau est détaillée sur le schéma de la Figure 3.23 . On commence par créer le réseau en fonction des hyperparamètres précisés par l'utilisateur, de la longueur des trames de la trace, et de leur contenu. Suite à quoi, les paramètres du réseau sont initialisés à l'aide des données de l'analyse statistique précédemment effectuée. Ensuite, vient l'étape d'apprentissage des paramètres du réseau à l'aide de l'EM, lors de laquelle on effectue une passe d'inférence classique, puis on calcule l'Estimation de la Statistique Suffisante (ESS) (on expliquera cette notion dans la section 3.3.3) et l'espérance de la log-vraisemblance des observations sachant les paramètres actuels du modèle. On met ensuite à jour les paramètres

du réseau, puis on regarde si le critère d'arrêt, à savoir la variation de l'espérance de la log-vraisemblance d'une itération à la suivante est vérifié ou non pour déterminer si l'apprentissage doit être poursuivi. Enfin, le découpage le plus probable est déterminé via une inférence de Viterbi permettant de trouver l'état du réseau le plus probable, puis par simple déduction des champs.

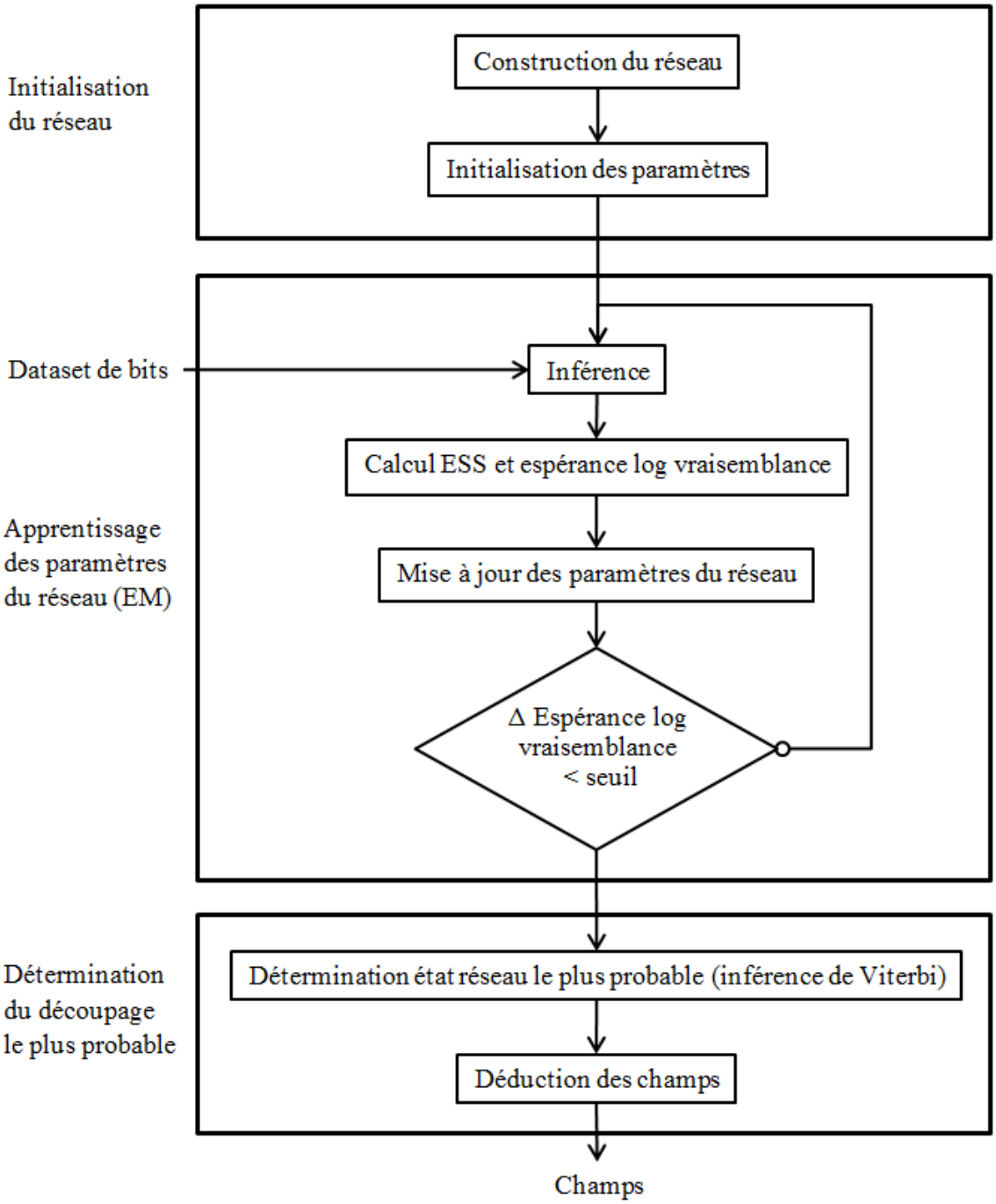


FIGURE 3.23 – Vue globale de l'opération d'analyse effectuée par le modèle

3.3.2 Réseau Bayésien employé

Le réseau employé par le modèle est représenté sur la Figure 3.24. Ce réseau sert à modéliser une trame d'un protocole binaire ; il s'agit donc d'un modèle génératif qu'on utilise pour trouver la structure sous-jacente des trames à analyser.

Comme on peut le voir, un motif se répète clairement (seul deux ont été représentés pour éviter la surcharge), comme dans un réseau Bayésien dynamique. Chaque motif caractérise le comportement d'une trame à une position donnée en terme d'octet ou de bit. Cependant, chacune des six variables de chacun des motifs possède son propre jeu de paramètres indépendants des autres, comme dans un réseau Bayésien. Un tel choix a été fait car une trame est à la fois une suite de bits (ou octets), donc présente une cyclicité au niveau du bit (ou de l'octet), et présente aussi, dans le cas d'un protocole binaire, des champs fortement dépendants de leur position. La nature récurrente du réseau Bayésien dynamique est donc utilisée pour traduire la cyclicité de la trame, et l'indépendance des paramètres d'un motif à l'autre permet un apprentissage indépendant du comportement à chaque position de la trame. Cela permet également quelques optimisations, en ne créant que les liens et les états de variables nécessaires à chaque position.

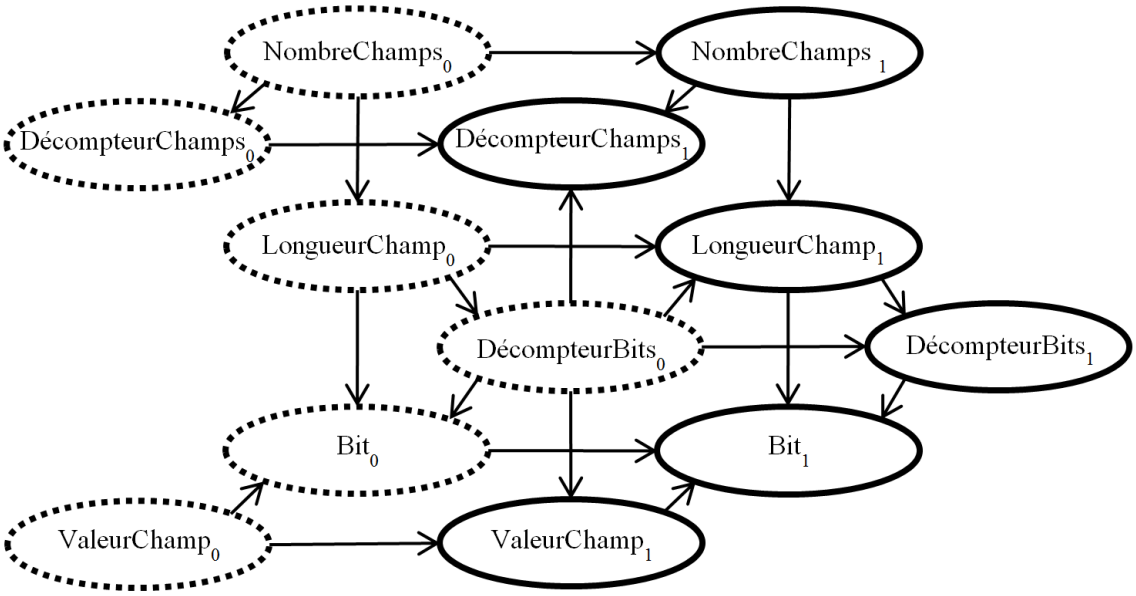


FIGURE 3.24 – Réseau Bayésien utilisé dans le modèle

Nous allons maintenant présenter le rôle de chaque variable dans un motif, ainsi que ses caractéristiques.

Le nœud **NombreChamps** donne le nombre de champs que contient la trame. Il s'agit d'un nœud aléatoire à la première position, et déterministe ensuite. À la première position de la trame, sa valeur est tirée, et il conserve ensuite la même valeur d'une position à l'autre avec une probabilité de 1. Les paramètres non-déterministes de ce nœud sont mis à jour durant l'apprentissage. À l'initialisation de l'apprentissage, les différents nombres de champs possibles

sont considérés équiprobables. Dans les équations subséquentes, on désignera ce nœud par N.

Le nœud **DecompteurChamp** donne le nombre de champs restant dans la trame à la position actuelle. Une valeur de 0 signifie que l'on se situe au dernier champ de la trame. Il s'agit d'un nœud déterministe. À la première position de la trame, il prend la valeur de NombreChamps avec une probabilité de 1, puis se décrémente de 1 à chaque fois qu'un champ se termine (la survenue de la fin d'un champ est su grâce à la valeur de DecompteurBit (ou DecompteurOctet) à la position précédente) avec une probabilité de 1. Les tables de probabilités conditionnelles de ce nœud ne sont pas mises à jour lors de l'apprentissage. On considère sa valeur observée à 0 à l'extrémité de la trame. Dans les équations subséquentes, on désignera ce nœud par DC.

Le nœud **LongueurChamp** donne la longueur du champ dans lequel se situe le bit (ou octet) actuel. Il s'agit d'un nœud semi-déterministe. Si l'on se situe au premier bit (ou octet) d'un champ (la survenue de la fin d'un champ est su grâce à la valeur de DecompteurBit (ou DecompteurOctet) à la position précédente), il détermine la probabilité que le nouveau champ soit d'une certaine longueur sachant la position actuelle (implicitement contenue dans la variable elle-même) et la longueur du champ précédent. Si l'on se situe au sein d'un champ, il prend la même valeur qu'à la position précédente avec une probabilité de 1, quelle que soit la position. Les parties non-déterministes des tables de probabilités conditionnelles de ce nœud sont mises à jour durant l'apprentissage. À l'initialisation de l'apprentissage, pour chaque position, et quelle que soit la longueur du champ précédent, on considère les différentes longueurs possibles du nouveau champ comme équiprobables. Ce nœud dépend également de NombreChamps afin de pouvoir compenser le fait que plus le nombre de transitions d'un champ à un autre est important, plus la probabilité de la trame est faible (multiplications successives par un nombre inférieur à 1). Ainsi les probabilités de transitions sont augmentées lorsque le nombre de champs est plus important, de façon à ce que le produit de toutes les probabilités de transition soit toujours égal à la même valeur, quel que soit le nombre de champs considéré de la trame. Cette compensation induit la création d'un état défaut, dont on ajuste la probabilité pour conserver la somme des probabilités des états possibles du nœud sachant ses parents égale à 1. Cet état est rendu impossible à tirer en spécifiant, pour un nœud observable dépendant de LongueurChamp, que sa valeur est forcée à une valeur jamais observée si LongueuChamp se trouve dans l'état défaut. La longueur maximale que permet ce nœud est définie de telle sorte que la trace à analyser sera forcément divisée en au moins deux champs, quelle que soit la passe considérée. Dans les équations subséquentes, on désignera ce nœud par L.

Le nœud **DecompteurBit** (ou **DecompteurOctet**) donne le nombre de bits (ou octets) restant dans le champ en cours à la position actuelle. Une valeur de 0 signifie que l'on se situe au dernier bit (ou octet) du champ en cours. Il s'agit d'un nœud déterministe. Si l'on se situe au début d'un champ, sa valeur est celle de LongueurChamp avec une probabilité de 1, sinon, sa valeur se décrémente de 1 à chaque position successive avec une probabilité de 1. Les tables de probabilités conditionnelles de ce nœud ne sont pas mises à jour lors de l'apprentissage. On considère sa valeur observée à 0 à l'extrémité de la trame. Dans les équations subséquentes, on désignera ce nœud par DB (ou DO).

Le nœud **ValeurChamp** donne une valeur n'ayant aucun sens en elle-même, mais servant plutôt à donner un degré de liberté au modèle pour créer une dépendance entre les champs, via

un "ID" commun qui n'est autre qu'un état de ce nœud. Il s'agit d'un nœud semi-déterministe. Si l'on se situe au premier bit (ou octet) d'un champ (la survenue de la fin d'un champ est su grâce à la valeur de DecompteurBit (ou DecompteurOctet) à la position précédente), il tire une nouvelle valeur. Si l'on se situe au sein d'un champ, il prend la même valeur qu'à la position précédente avec une probabilité de 1. Les parties non-déterministes des tables de probabilités conditionnelles de ce nœud sont mises à jour durant l'apprentissage. À l'initialisation de l'apprentissage, les probabilités d'obtenir chaque valeur possible à un changement de champ sont tirées aléatoirement. Dans les équations subséquentes, on désignera ce nœud par V.

Le nœud **Bit** (ou **Octet**) est le nœud observé du réseau, donnant les bits (ou octets) successifs des trames. Il s'agit d'un nœud aléatoire dépendant de la longueur du champ actuel, de la position dans ce champ, de l'"ID" du champ donné par ValeurChamp, et d'un certain nombre de bits (ou octets) précédents. Les tables de probabilités conditionnelles de ce nœud sont mises à jour lors de l'apprentissage, et sont initialisées à l'aide des statistiques extraites du dataset d'apprentissage (trace). Dans les équations subséquentes, on désignera ce nœud par B (ou O).

Dans la suite de ce chapitre, on considérera uniquement le cas dans lequel la granularité est située au niveau du bit, mais le raisonnement reste valable à l'échelle de l'octet, à ceci près qu'au lieu de deux valeurs possibles, il y en a 256. On utilisera la représentation du réseau de la Figure 3.25 comme référence pour le reste de ce chapitre.

L'ensemble X des nœuds du réseau pour une trame m est donc le suivant :

$$X = \{N_t, DC_t, L_t, DV_t, V_t, B_t\}_{t\in[1,T_m]}, \tag{3.21}$$

où l'indice t représente la position à laquelle est considéré le nœud, et T_m la longueur de la trame m.

Le réseau se basant sur la recherche de l'enchaînement de séquences le plus probable, il doit avoir accès à la probabilité d'une séquence de bits, cependant, la granularité du réseau étant située au niveau du bit, il convient d'effectuer une décomposition de la probabilité de la séquence, donc de la probabilité conjointe des bits la composant, en terme de probabilités marginales et conditionnelles des bits. Pour cela, on applique la formule suivante, dérivée du théorème de Bayes, à une séquence S de N bits :

$$\mathbb{P}(S_1^N) = \mathbb{P}(\bigcap_{t=1}^{N} B_t) = \prod_{t=1}^{N} \mathbb{P}(B_t|\bigcap_{i=1}^{t-1} B_i) \tag{3.22}$$

avec t la position du bit dans la séquence.

On connaît désormais les probabilités conditionnelles nécessaires au calcul des probabilités des séquences. Pour calculer les probabilités conditionnelles à partir des statistiques des séquences, on utilise directement la formule de Bayes :

$$P(B_t|\bigcap_{i=1}^{t-1} B_i) = \frac{\mathbb{P}(\bigcap_{i=1}^{t} B_i)}{\mathbb{P}(\bigcap_{i=1}^{t-1} B_i)} \tag{3.23}$$

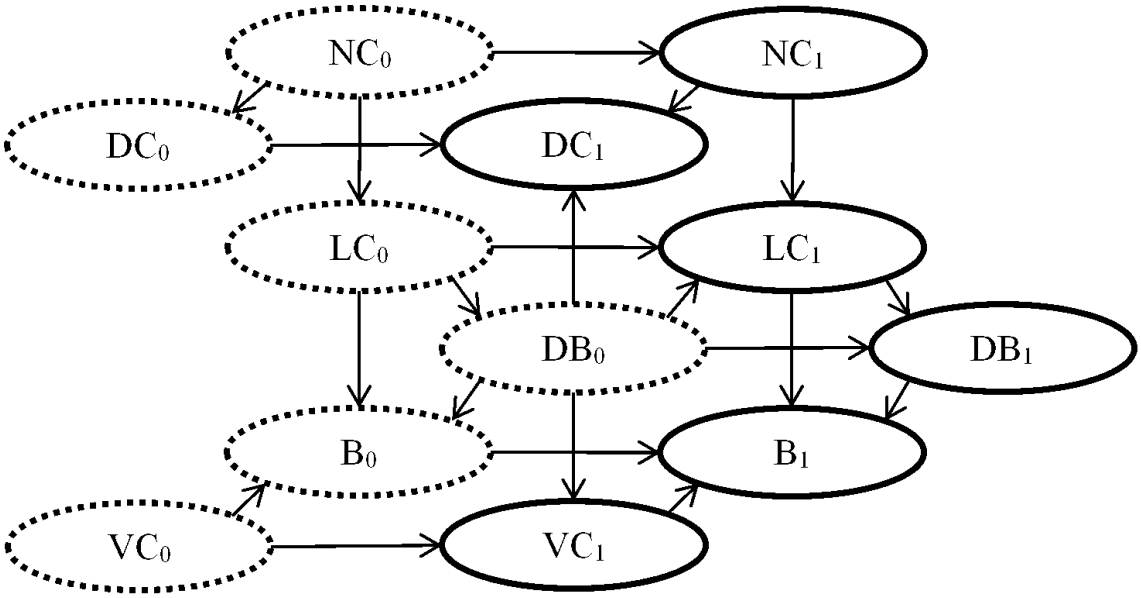


FIGURE 3.25 – Réseau de référence

L'horizon maximal (ordre maximal) auquel un nœud Bit peut dépendre de la valeur des bits précédents est considéré égal à la longueur maximale des champs que l'on cherche, spécifié par la plus grande longueur tirable à l'aide du nœud LongueurChamp. Cependant, pour maîtriser la complexité du réseau, on ajoute un hyperparamètre permettant de définir un horizon maximal plus faible. Également afin de maîtriser la complexité, lorsque la granularité considérée est l'octet, on ne considère qu'un nombre limité de valeurs possibles, celles les plus présentes dans le dataset, et toutes les autres valeurs sont remplacées par une valeur par défaut. Le nombre maximal de valeurs différentes de l'observation considéré est défini par un hyperparamètre.

3.3.3 Apprentissage des paramètres à l'aide de l'EM

L'algorithme EM se déroule en trois phases, répétées circulairement jusqu'à vérification du critère d'arrêt :

- La phase ***Expectation***, dans laquelle on cherche, pour chaque trame, à inférer les probabilités conjointes de chaque nœud et ses parents, et ce pour chaque nœud. On dispose pour cela du réseau et de ses paramètres, ainsi que des observations dans le dataset d'apprentissage.
- Le calcul de **l'espérance de la log-vraisemblance** des observations, dont on utilise la variation de la valeur d'une itération d'EM à la suivante comme critère d'arrêt.
- La phase ***Maximization***, dans laquelle on cherche à maximiser l'espérance de la log-vraisemblance précédemment calculée en recalculant les paramètres optimaux du réseau. On dispose à cette fin des probabilités conjointes de chaque nœud et ses parents calculées dans la phase *Expectation*.

Nous présenterons tour à tour ces trois phases appliquées au réseau de BaNet3F.

Phase *Expectation*

La boucle de l'EM commence par la phase *Expectation*, dans laquelle on calcule les probabilités conjointes nécessaires à la mise à jour des paramètres. La façon la plus directe de le faire consiste à marginaliser la probabilité conjointe de tous les nœuds du réseau pour une trame m :

$$\begin{aligned} P(\bigcap_{t=0}^{T_m-1} N_t, DC_t, L_t, DB_t, V_t, B_t) = & \prod_{t=0}^{T_m-1} P(N_t|N_{max(0,t-1)}) \\ & \times (\mathbb{1}_{t\geq 1} P(DC_t|DC_{t-1}, DB_{t-1}) + \mathbb{1}_{t=0} P(DC_t|N_t)) \\ & \times (\mathbb{1}_{t\geq 1} P(L_t|L_{t-1}, N_t, DB_{t-1}) + \mathbb{1}_{t=0} P(L_t|N_t)) \\ & \times P(DB_t|DB_{max(0,t-1)}, L_t) \\ & \times (\mathbb{1}_{t\geq 1} P(V_t|V_{t-1}, DB_{t-1}) + \mathbb{1}_{t=0} P(V_t)) \\ & \times P(B_t|L_t, DB_t, V_t, \bigcap_{j=max(0,t-horizon)}^{t-1} B_j), \end{aligned} \tag{3.24}$$

où $\mathbb{1}_x$ est la fonction indicatrice, valant 1 lorsque la condition x est vérifiée, et 0 sinon. L'horizon quantifie le nombre maximal de positions de bit que l'on considère simultanément.

Bien entendu, la taille de la table des probabilités conjointes du réseau explosant exponentiellement, on n'utilise pas cette méthode pour effectuer l'inférence, mais l'algorithme de Lauritzen [69] basé sur les arbres à jonctions et le passage de message. Cette dernière approche permet de ne jamais considérer la table conjointe globale du réseau, mais une multitude de petites tables locales, empêchant l'explosion de la mémoire et du temps de calcul nécessaires.

Calcul de l'espérance de la log-vraisemblance

On calcule ensuite l'espérance de la log-vraisemblance des observations sachant le réseau, qui correspond à la fonction Q présentée dans la section 3.1.4. Dans le cas du réseau de notre modèle, Q est définie comme suit :

$$Q(\Theta, \Theta^r) = Q(\Theta, \Theta^r)_0 + \sum_{t=1}^{T_m-1} Q(\Theta, \Theta^r)_t, \tag{3.25}$$

où $Q(\Theta, \Theta^r)_t$ est la partie de la valeur de Q concernée par les variables à la position t. $Q(\Theta, \Theta^r)_0$ est défini par :

$$\begin{aligned} Q(\Theta, \Theta^r)_0 = & \sum_{k_N} logP(N_0 = k_N) \sum_{m=1}^{M} P(N_0^m = k_N|{}^oX^m, \Theta^r) \\ & + \sum_{k_N, k_{DC}} logP(DC_0 = k_{DC}|N_0 = k_N) \sum_{m=1}^{M} P(DC_0^m = k_{DC}, N_0^m = k_N|{}^oX^m, \Theta^r) \\ & + \sum_{k_N, k_L} logP(L_0 = k_L|N_0 = k_N) \sum_{m=1}^{M} P(L_0^m = k_L, N_0^m = k_N|{}^oX^m, \Theta^r) \end{aligned} \tag{3.26}$$

$$
\begin{aligned}
&+ \sum_{k_L,k_{DB}} logP(DB_0 = k_{DB}|L_0 = k_L) \sum_{m=1}^{M} P(DB_0^m = k_{DB}, L_0^m = k_L|{}^oX^m, \Theta^r) \\
&+ \sum_{k_V} logP(V_0 = k_V) \sum_{m=1}^{M} P(V_0^m = k_V|{}^oX^m, \Theta^r) \\
&+ \sum_{k_L,k_{DB},k_V,k_B} logP(B_0 = k_B|L_0 = k_L, DB_0 = k_{DB}, V_0 = k_V) \\
&\times \sum_{m=1}^{M} P(B_0^m = k_B, L_0^m = k_L, DB_0^m = k_{DB}, V_0^m = k_V|{}^oX^m, \Theta^r).
\end{aligned}
$$

$Q(\Theta, \Theta^r)_t$ est défini par :

$$
\begin{aligned}
Q(\Theta, \Theta^r)_t = &\sum_{k_N^{t-1},k_N^t} logP(N_t = k_N^t|N_{t-1} = k_N^{t-1}) \sum_{m=1}^{M} P(N_t^m = k_N^t, N_{t-1}^m = k_N^{t-1}|{}^oX^m, \Theta^r) \\
&+ \sum_{k_{DB},k_{DC}^{t-1},k_{DC}^t} logP(DC_t = k_{DC}^t|DC_{t-1} = k_{DC}^{t-1}, DB_{t-1} = k_{DB}) \\
&\times \sum_{m=1}^{M} P(DC_t^m = k_{DC}^t, DC_{t-1}^m = k_{DC}^{t-1}, DB_{t-1}^m = k_{DB}|{}^oX^m, \Theta^r) \\
&+ \sum_{k_N,k_{DB},k_L^{t-1},k_L^t} logP(L_t = k_L^t|L_{t-1} = k_L^{t-1}, N_t = k_N, DB_{t-1} = k_{DB}) \\
&\times \sum_{m=1}^{M} P(L_t^m = k_L^t, L_{t-1}^m = k_L^{t-1}, N_t^m = k_N, DB_{t-1}^m = k_{DB}|{}^oX^m, \Theta^r) \\
&+ \sum_{k_L,k_{DB}^{t-1},k_{DB}^t} logP(DB_t = k_{DB}^t|DB_{t-1} = k_{DB}^{t-1}, L_t = k_L) \\
&\times \sum_{m=1}^{M} P(DB_t^m = k_{DB}^t, DB_{t-1}^m = k_{DB}^{t-1}, L_t^m = k_L|{}^oX^m, \Theta^r) \\
&+ \sum_{k_{DB},k_V^{t-1},k_V^t} logP(V_t = k_V^t|V_{t-1} = k_V^{t-1}, DB_{t-1} = k_{DB}) \\
&\times \sum_{m=1}^{M} P(V_t^m = k_V^t, V_{t-1}^m = k_V^{t-1}, DB_{t-1}^m = k_{DB}|{}^oX^m, \Theta^r) \\
&+ \sum_{k_L,k_{DB},k_V,\{k_B^i\}_{i\in I}} logP(B_t = k_B^t|L_t = k_L, DB_t = k_{DB}, V_t = k_V, \bigcap_{j\in I}(B_j = k_B^j)) \\
&\times \sum_{m=1}^{M} P(B_t^m = k_B^t, L_t^m = k_L, DB_t^m = k_{DB}, V_t^m = k_V, \bigcap_{j\in I}(B_j^m = k_B^j)|{}^oX^m, \Theta^r).
\end{aligned}
\tag{3.27}
$$

Avec ${}^oX^m = \{{}^oB_t^m\}_{t\in[1,T_m]} \cup^o DC_{T_m^m} \cup^o DB_{T_m^m}$, les observations de la trame m, où $\{{}^oB_t^m\}_{t\in[1,T_m]}$ représente les valeurs observées de l'ensemble des bits de la trame m, ${}^oDC_{T_m^m}$ représente la valeur observée de DecompteurChamp au dernier bit de la trame, et ${}^oDB_{T_m^m}$ représente la valeur observée de DecompteurBit au dernier bit de la trame. k_i^t et k_i^{t-1} sont deux compteurs d'états de la variable i évoluant indépendamment. I est l'ensemble des positions de bits considérées simultanément lorsque l'on se situe à la position t : $[max(0, t-horizon), t-1]$.

Phase *Maximization*

On achève la boucle de l'EM par la phase *Maximization*, lors de laquelle chacun des paramètres de chacune des variables du réseau est mis à jour afin de maximiser Q. On prendra l'exemple de la variable LongueurChamp à la position t, L_t, pour illustrer cette opération, car toutes les autres variables sont mises à jour selon le même principe. On pose $\Theta^{r+1}_{L_t,j_{L_t},k_{L_t}}$, la probabilité conditionnelle du réseau mise à jour du nœud L_t lorsqu'il vaut k_{L_t} et ses parents j_{L_t}, définie comme suit :

$$\Theta^{r+1}_{L_t,j_{L_t},k_L} = \frac{\widehat{N_{L_t,j_{L_t},k_L}}}{\sum\limits_{k_L} \widehat{N_{L_t,j_{L_t},k_L}}} \tag{3.28}$$

$\widehat{N_{L_t,j_{L_t},k_L}}$ est l'Estimation de la Statistique Suffisante (l'Expected Sufficient Statistic ou ESS) de la probabilité conjointe du nœud L_t et de ses parents, lorsqu'il vaut k_L et ses parents j_{L_t}, c'est à dire la probabilité estimée lors de l'inférence à l'aide du réseau et ses paramètres Θ^r et des observations. L'ESS est calculée comme suit, à partir des probabilités conjointes trouvées lors de la phase *Expectation* :

— Pour la position 0 :

$$\widehat{N_{L_0,j_{L_0},k_L}} = \sum_{m=1}^{M} P(L_0^m = k_L, N_0^m = j_{L_0}^{N_0} | {}^oX^m, \Theta^r), \tag{3.29}$$

où j_x^y représente la valeur de la variable y dans la combinaison de valeur des parents de x.

— Pour les positions de 1 à T_m :

$$\widehat{N_{L_t,j_{L_t},k_L}} = \sum_{m=1}^{M} P(L_t^m = k_L, L_{t-1}^m = j_{L_t}^{L_{t-1}}, N_t^m = j_{L_t}^{N_t}, DB_{t-1}^m = j_{L_t}^{DB_{t-1}} | {}^oX^m, \Theta^r). \tag{3.30}$$

3.3.4 Détermination du découpage le plus probable à l'aide de l'algorithme AMBV

Présentation de l'algorithme

Une fois l'apprentissage terminé, l'état global du réseau le plus probable est inféré à l'aide de l'algorithme de Viterbi présenté dans la section 3.1.3. Comme cela a été mentionné à cet endroit, le principal levier d'optimisation de cet algorithme repose sur le choix de la suite des frontière de Markov permettant de parcourir toutes les variables du réseau. Afin de pouvoir rapidement changer le réseau utilisé dans le modèle lors de sa phase de développement, nous avons conçu un algorithme permettant de construire automatiquement cette suite de frontières

à partir d'un réseau Bayésien quelconque. Il apporte également diverses autres optimisations permettant d'accélérer l'inférence de Viterbi pour chaque trame de la trace. Cet algorithme ne prétend pas trouver les meilleures frontières de Markov possibles, seulement un bon jeu de frontières assurant le succès du traitement de Viterbi, de façon simple et rapide. Le gain de cet algorithme est d'autant plus important que le nombre d'inférence à faire avec le même réseau l'est.

Le déroulement global de l'algorithme est le suivant :

1. On forme la frontière de Markov actuelle de départ (FMI) en y plaçant les variables ne possédant aucun parent.
2. Tant qu'il y a des nœuds dans la frontière de Markov actuelle (FMA) :
   (a) On optimise l'ordre des nœuds dans l'ensemble constitué de la frontière de Markov actuelle et la précédente, afin de pouvoir sauter certains sous-ensembles d'états lors du balayage de toutes les transitions possibles.
   (b) On calcule les probabilités de transitions de chaque état de la frontière de Markov précédente (FMP) (si elle existe) vers chaque état de la frontière de Markov actuelle, en ne considérant aucune observation, et on stocke chaque transition possible (probabilité supérieure à 0), ainsi que sa probabilité associée. Cela permet de réduire au maximum les transitions considérées d'une frontière de Markov à la suivante.
   (c) On enregistre également les états de la frontière de Markov actuelle auxquels il est possible d'accéder, afin de réduire les états considérés pour chaque frontière de Markov aux itérations suivantes.
   (d) Chaque nœud de la frontière de Markov actuelle vote pour ses enfants, et chaque enfant vote à son tour pour ses propres enfants.
   (e) Tous les nœuds possédant au moins un parent dans la frontière de Markov actuelle, ayant reçu au moins un vote de chacun de leurs parents, et ne faisant pas partie de la frontière de Markov actuelle intègrent la frontière de Markov suivante (FMS).
   (f) Ensuite, si tous les enfants d'un nœud de la frontière de Markov actuelle ne sont pas compris dans la frontière de Markov suivante, la frontière de Markov actuelle, ou l'une des frontières de Markov précédentes, on met ce nœud dans la frontière de Markov suivante.
   (g) La frontière de Markov actuelle devient la frontière de Markov précédente et la frontière de Markov suivante devient la frontière de Markov actuelle.
3. Enfin, pour chaque réalisation du réseau (trame) dans le dataset, on utilise les données stockées pour effectuer l'inférence de Viterbi en elle-même en un minimum d'opérations, et ainsi déterminer le chemin le plus probable sachant les observations.

L'algorithme est présenté de façon formelle dans les Algorithmes 9,10, et 11.

On précise que $Etats(FMCO)$ est parcouru en incrémentant de manière systématique les variables de $FMCO$ dans l'ordre croissant. Cela signifie qu'on balaie d'abord toutes les valeurs de $FMCO_1$ avec toutes les autre variables de $FMCO$ à leur première valeur, avant de recommencer avec cette fois-ci $FMCO_2$ à sa seconde valeur. On continue avec toutes les

Algorithme 9 : AMBV - Initialisation

Données : réseau R, ensemble des observations O
Résultat : État le plus probable du réseau S et sa probabilité P_{max}

```
pour {i ∈ R/par(X_i) = ∅} faire
    FM_1 = FM_1 ∪ X_i;
fin
```

valeurs de $FMCO_2$ avant de passer à le seconde valeur de $FMCO_3$, et ainsi de suite. Cette logique de balayage est appliquée constamment, même lorsque les valeurs des $FMCO_i$ sont modifiées au sein de la boucle.

Les états sautés dans la boucle $e \in Etats(FMCO)$ comptent comme balayés, et donc contribuent à la validation du critère de fin de boucle.

X_i représente la $i^{ème}$ variable du réseau, $par(v)$ représente l'ensemble des parents de v, et e_k désigne l'état de la variable k dans l'état e de la frontière de Markov.

E est la structure stockant les états possibles de chaque frontière de Markov. Ainsi, E_{i-1} contient tous les états possibles de la frontière de Markov $i-1$

$P^e_{FM_i}$ est une variable calculée progressivement, représentant la probabilité de la frontière de Markov i sachant l'état considéré e de $FMCO$.

$Card(FM_i)$ représente le nombre de variables de FM_i, $FMCO_k$ représente la $k^{ème}$ variable des frontières de Markov précédente et actuelle ordonnées, et $e_{FMCO_1^{Card(FM_i)}}$ désigne les états dans e des variables de $FMCO$ de 1 à $Card(FM_i)$.

T est une structure tridimensionnelle contenant les données de transition, dont la première dimension est caractérisée par les deux frontières de Markov concernées par la transition, la deuxième dimension est caractérisée par l'état cible de la transition, et la troisième dimension par l'état source. Ainsi, ${}^{i}_{i-1}T^{e_{FMCO_1^{Card(FM_i)}}}_{e_{FMCO^{Card(FMCO)}_{Card(FM_i)+1}}} = P^e_{FM_i}$ est l'élément de la structure T contenant la probabilité de la transition de l'état $e_{FMCO^{Card(FMCO)}_{Card(FM_i)+1}}$ de la frontière de Markov $i-1$ à l'état $e_{FMCO_1^{Card(FM_i)}}$ de la frontière de Markov i.

$enf(j)$ représente les enfants de la variable j, et $V(k,j)$ représente le nombre de votes pour la variable k par la variable j.

$Card(FM)$ est le nombre de frontières de Markov du réseau, O_v désigne l'observation de la variable v, $t(cible)$ est l'état cible de la transition t, et $t(cible)_v$ désigne l'état de la variable v dans $t(cible)$.

L'algorithme est également résumé sous forme d'algorigramme Figure 3.26.

Illustration de la génération des frontières de Markov

Nous allons maintenant appliquer une partie de l'algorithme présenté à une version réduite du réseau de BaNet3F, afin de montrer comment les frontières de Markov sont générées. La Figure 3.27 illustre graphiquement les éléments principaux de la description du mécanisme qui suit. Chaque couleur représente une frontière de Markov, et les numéros indiquent la position du noeud dans la frontière de Markov concernée.

Algorithme 10 : AMBV - Génération de la structure d'optimisation

$i = 1$;
tant que $FM_i \neq \emptyset$ **faire**
  $FMC = FM_i \cup \{v \in FM_{i-1}/v \notin FM_i\}$;
  pour $\{j \in FMC/\nexists v \in FMC/j \in par(v)\}$ **faire**
    $FMCO = FMCO \cup j$; (Ordre d'ajout des variables fondamental)
    $FMC = FMC \backslash j$;
    j est réinitialisé;
  fin
  pour $e \in Etats(FMCO)$ **faire**
    $Interruption = 0$;
    si $i > 1$ *et* $e_{FMCO_{Card(FM_i)+1}^{Card(FMCO)}} \notin E_{i-1}$ **alors**
      $Interruption = Card(FM_i) + 1$;
    sinon
      $P^e_{FM_i} = 1$;
      pour $k = Card(FM_i)$ *à 1 par pas de -1* **faire**
        si $P(FMCO_k = e_{FMCO_k} | par(FMCO_k) = e_{par(FMCO_k)}) \neq 0$ **alors**
          $P^e_{FM_i} = P^e_{FM_i} P(FMCO_k = e_{FMCO_k} | par(FMCO_k) = e_{par(FMCO_k)})$;
        sinon
          $Interruption = k$;
          Interruption de la boucle sur k;
        fin
      fin
    fin
    si $Interruption = 0$ **alors**
      ${}^{i}_{i-1}T^{e_{FMCO_1^{Card(FM_i)}}}_{e_{FMCO_{Card(FM_i)+1}^{Card(FMCO)}}} = P^e_{FM_i}$;
      On ajoute $e_{FMCO_1^{Card(FM_i)}}$ à E_i
    sinon
      $e_{FMCO_{Interruption}} + +$;
      $e_{FMCO_1} = 0$;
    fin
  fin
  pour $j \in FM_i$ **faire**
    pour $k \in enf(j)$ **faire**
      $V(k, j) + +$;
      pour $l \in enf(k)$ **faire**
        $V(l, k) + +$;
      fin
    fin
  fin
  pour $\{j \in R \backslash \bigcup_{n=1}^{i} FM_n / \exists u \in par(j)/u \in FM_i \ et \ \forall v \in par(j), V(j, v) > 0\}$ **faire**
    $FM_{i+1} = FM_{i+1} \cup j$;
  fin
  pour $\{j \in FM_i / \exists v \in enf(j)/v \notin \bigcup_{n=1}^{i+1} FM_n\}$ **faire**
    $FM_{i+1} = FM_{i+1} \cup j$;
  fin
  $i + +$;
fin

Algorithme 11 : AMBV - Inférence de Viterbi

pour $o = 1$ *à* $Card(O)$ **faire**
  pour $i = 1$ *à* $Card(FM)$ *par pas de 1* **faire**
    pour $t \in T^i_{i-1}$ **faire**
      si $\nexists v \in FM_i / O^v_o \neq t(cible)_v$ **alors**
        $P(FM_i^{t(cible)}) = \max_{k \in {}^{t(cible)}T^i_{i-1}} P(FM^k_{i-1}) \, {}^{t(cible)}_{k}T^i_{i-1}$;
        $A_i^{t(cible)} = \operatorname{argmax}_{k \in {}^{t(cible)}T^i_{i-1}} P(FM^k_{i-1}) \, {}^{t(cible)}_{k}T^i_{i-1}$;
      sinon
        $P(FM_i^{t(cible)}) = 0$;
        $A_i^{t(cible)} = 0$;
      fin
    fin
  fin
  $P_{max} = \max_{i \in FM_{Card(FM)}} P(FM^i_{Card(FM)})$;
  $S_{Card(FM)} = \operatorname{argmax}_{i \in FM_{Card(FM)}} P(FM^i_{Card(FM)})$;
  pour $i = Card(FM)$ *à* 2 *par pas de* -1 **faire**
    $S_{i-1} = A_i^{S_i}$;
  fin
fin

Tout d'abord, on place dans la frontière de Markov 1 les variables $NombreChamps_0$ et $ValeurChamp_0$, car ce sont les seules dénuées de parents.

Ensuite, l'étape d'ajout des variables de la frontière précédente est une non-opération car la frontière n'existe pas. De même, les variables n'ayant pas d'enfants dans la frontière 1, elles ne sont pas réordonnées.

La variable $NombreChamps_0$ vote pour ses enfants $DecompteurChamps_0$, $LongueurChamp_0$, et $NombreChamps_1$, puis $DecompteurChamps_0$ vote pour son propre enfant $DecompteurChamps_1$, $LongueurChamp_0$ vote pour $DecompteurBits_0$, Bit_0 et $LongueurChamp_1$, et $NombreChamps_1$ vote pour $DecompteurChamps_1$ et $LongueurChamp_1$. $ValeurChamp_0$ vote pour Bit_0 et $ValeurChamp_1$, puis Bit_0 vote pour Bit_1, et $ValeurChamp_1$ vote pour Bit_1.

$DecompteurChamps_0$, de même que $LongueurChamp_0$ et $NombreChamps_1$ ont reçu un vote de leur unique parent, qui se trouve dans la frontière de Markov 1 ; on les inclut donc dans la frontière de Markov 2. $DecompteurChamps_1$ et $DecompteurBits_0$ ont bien reçu des votes de tous leurs parents, mais aucun de ceux-ci n'appartiennent à la frontière de Markov 1 ; on ne les inclut donc pas dans la frontière de Markov 2. Les variables Bit_0, $LongueurChamp_1$, $ValeurChamp_1$, et Bit_1 n'ont pas reçu un vote de chacun de leurs parent ; elles ne sont donc pas non plus incluses dans la frontière de Markov 2.

Les variables Bit_0 et $ValeurChamp_1$ n'étant pas incluses dans les frontières de Markov 1

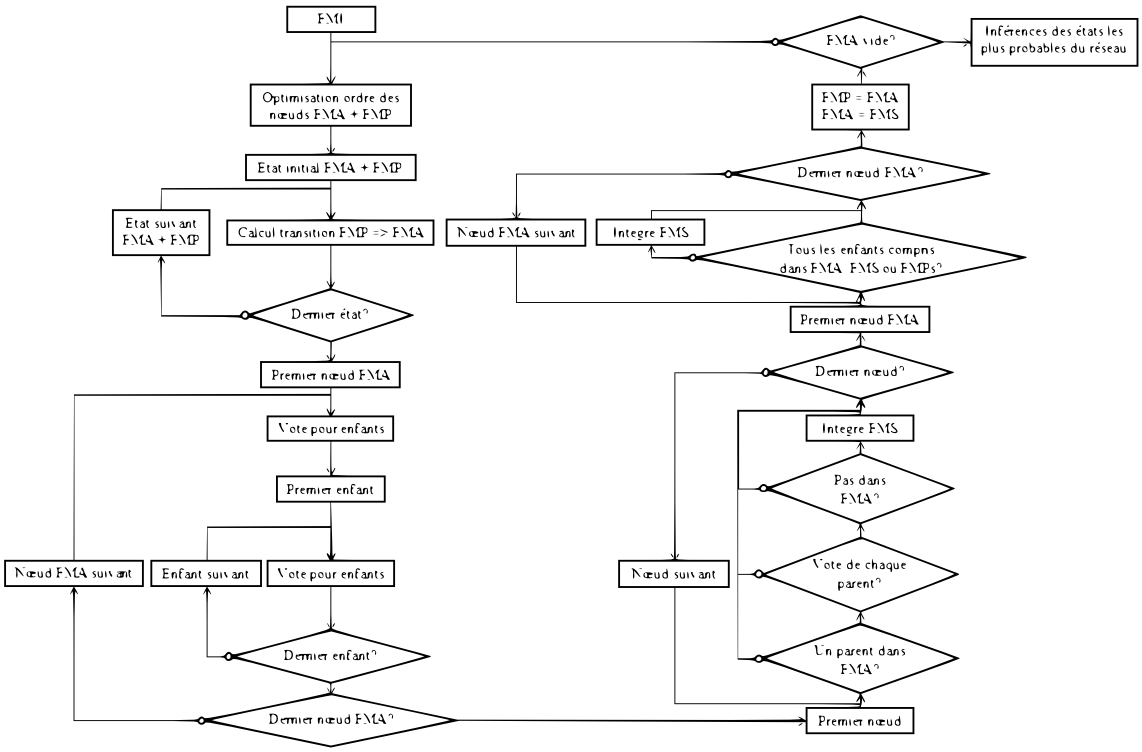


FIGURE 3.26 – Algorigramme du déroulement général de l'algorithme AMBV

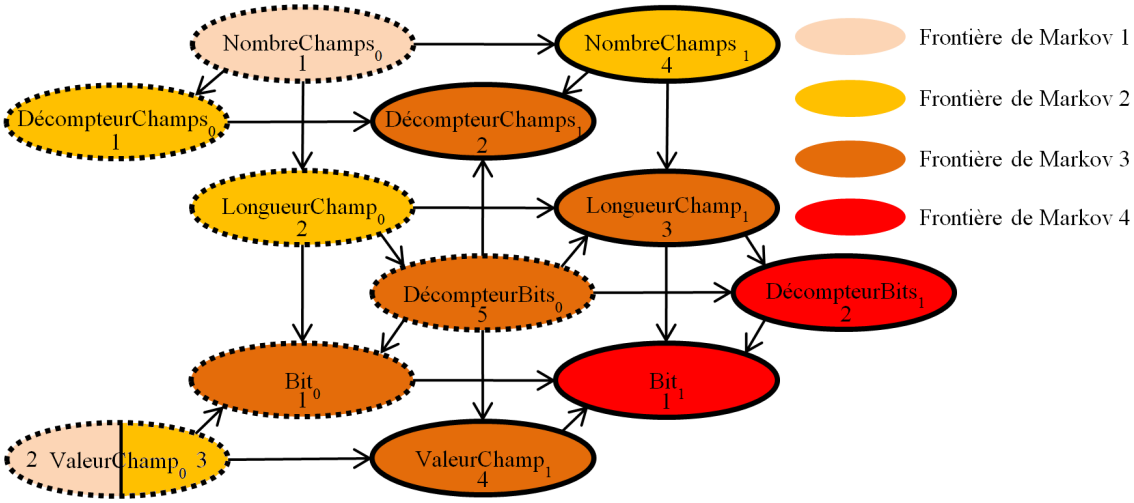


FIGURE 3.27 – Frontières de Markov du réseau de BaNet3F générées avec l'algorithme

ou 2, la variable $ValeurChamp_0$ doit être également insérée dans la frontière de Markov 2.

La frontière de Markov 2 contient donc les variables $DecompteurChamps_0$, $LongueurChamp_0$, $ValeurChamp_0$, et $NombreChamps_1$.

Aucunes de ces variables ne possédant d'enfant dans la frontière de Markov 2, elles ne

sont pas réordonnées.

La variable $DecompteurChamps_0$ vote pour son enfant $DecompteurChamps_1$, qui ne vote pas car il n'a pas d'enfants. $LongueurChamp_0$ vote pour $LongueurChamp_1$, $DecompteurBits_0$ et Bit_0, puis $LongueurChamp_1$ vote pour $DecompteurBits_1$ et Bit_1, $DecompteurBits_0$ vote pour Bit_0, $DecompteurChamps_1$, $LongueurChamp_1$, $DecompteurBits_1$ et $ValeurChamp_1$, et Bit_0 vote pour Bit_1. $ValeurChamp_0$ vote pour Bit_0 et $ValeurChamp_1$, puis Bit_0 vote pour Bit_1, et $ValeurChamp_1$ vote pour Bit_1. $NombreChamps_1$ vote pour $DecompteurChamps_1$ et $LongueurChamp_1$, puis $DecompteurChamps_1$ ne vote pas car il n'a pas d'enfants, et $LongueurChamp_1$ vote pour $DecompteurBits_1$ et Bit_1.

$DecompteurBits_0$, Bit_0, $DecompteurChamps_1$, $LongueurChamp_1$, et $ValeurChamp_1$ ont reçu un vote de chacun de leurs parents, dont au moins un se trouve dans la frontière de Markov 2 ; on les inclut donc dans la frontière de Markov 3. $DecompteurBits_1$ a bien reçu des votes de tous ses parents, mais aucun de ceux-ci n'appartiennent à la frontière de Markov 2 ; on ne l'inclut donc pas dans la frontière de Markov 3. Bit_1 n'a pas reçu un vote de chacun de ses parents ; il n'est donc pas non plus inclus dans la frontière de Markov 3.

La frontière de Markov 3 contient donc les variables $DecompteurBits_0$, Bit_0, $DecompteurChamps_1$, $LongueurChamp_1$, et $ValeurChamp_1$.

On effectue ensuite un balayage systématique de toutes les variables de la frontière de Markov 3. La variable $DecompteurBits_0$ possédant 4 enfants dans la frontière de Markov 3, on la met de côté. Bit_0 ne possédant pas d'enfants dans la frontière de Markov 3, on la place en première position de la frontière de Markov 3 ordonnée, et on l'enlève de la frontière de Markov 3 d'origine. On réexamine ensuite $DecompteurBits_0$, qui ne possède désormais plus que 3 enfants dans la frontière de Markov 3 d'origine, et on le met de nouveau de côté. $DecompteurChamps_1$ ne possède pas d'enfants dans la frontière de Markov 3, on le place donc en seconde position de la frontière de Markov 3 ordonnée, et on l'enlève de la frontière de Markov 3 d'origine. $DecompteurBits_0$ possède toujours 2 enfants dans la frontière, on le met donc encore de côté, et on sélectionne $LongueurChamp_1$, qui ne possède pas d'enfants dans la frontière, qu'on ajoute à la suite dans la frontière ordonnée. $DecompteurBits_0$ possède toujours 1 enfant ; il est donc encore écarté, et on sélectionne $ValeurChamp_1$, qu'on place en quatrième position. Enfin, $DecompteurBits_0$ étant désormais seul dans la frontière, il ne possède plus d'enfants ; on l'ajoute donc en dernière position de la frontière ordonnée. La frontière est maintenant vide, la procédure d'ordonnancement prend donc fin.

La frontière de Markov 3, après avoir été ordonnée, contient désormais, dans l'ordre, Bit_0, $DecompteurChamps_1$, $LongueurChamp_1$, $ValeurChamp_1$, et $DecompteurBits_0$.

La variable Bit_0 vote pour son enfant Bit_1, qui ne vote pas car il n'a pas d'enfants. $DecompteurChamps_1$ ne vote pas car il n'a pas d'enfants. $LongueurChamp_1$ vote pour $DecompteurBits_1$ et Bit_1, puis $DecompteurBits_1$ vote pour Bit_1, et Bit_1 ne vote pas car il n'a pas d'enfants. $ValeurChamp_1$ vote pour Bit_1 , puis Bit_1 ne vote pas car il n'a pas d'enfants. $DecompteurBits_0$ vote pour Bit_0, $DecompteurChamps_1$, $LongueurChamp_1$, $DecompteurBits_1$ et $ValeurChamp_1$, puis Bit_0 vote pour Bit_1, $DecompteurChamps_1$ ne

vote pas car il n'a pas d'enfants, $LongueurChamp_1$ vote pour $DecompteurBits_1$ et Bit_1, $DecompteurBits_1$ vote pour Bit_1 et $ValeurChamp_1$ vote pour Bit_1.

$DecompteurBits_1$ et Bit_1 ont reçu au moins un vote de chacun de leurs parents, dont au moins un se trouve dans la frontière de Markov 3, on les inclut donc dans la frontière de Markov 4.

La frontière de Markov 4 contient donc les variables $DecompteurBits_1$ et Bit_1.

On effectue enfin un balayage systématique de toutes les variables de la frontière de Markov 4. La variable $DecompteurBits_1$ possédant 1 enfant dans la frontière de Markov 4, on la met de côté. Bit_1 ne possédant pas d'enfants dans la frontière de Markov 4, on la place en première position de la frontière de Markov 4 ordonnée, et on l'enlève de la frontière de Markov 4 d'origine. On réexamine ensuite $DecompteurBits_1$, qui ne possède désormais plus d'enfants dans la frontière de Markov 4 d'origine ; on le place donc en seconde position dans la frontière de Markov 4 ordonnée. La frontière d'origine étant maintenant vide, la procédure d'ordonnancement prend fin.

La frontière de Markov 4, après avoir été ordonnée, contient désormais, dans l'ordre, Bit_1 et $DecompteurBits_1$.
Aucune des variables de la frontière de Markov 4 n'ayant d'enfants, aucune variable ne peut être ajoutée à la frontière de Markov 5, qui se retrouve ainsi vide, mettant un terme à la génération des frontières de Markov.

3.3.5 Validation du découpage à l'aide de sa probabilité

Lorsque l'analyse à l'aide du réseau est terminée, on obtient un découpage D candidat sous la forme d'une suite ordonnée de champs caractérisés par leur longueur en bits, comme suit :

$$D = \{c_i, c_i \in \mathbb{N}^*\}_{i \in I}, \tag{3.31}$$

où I est l'ensemble des champs du découpage. Par exemple, on pourrait avoir un découpage {3,1,4,8,16,32}.

L'objectif est ensuite de déterminer si ce découpage est accepté ou non. Pour cela, à partir des statistiques du dataset d'apprentissage de l'itération actuelle du modèle, on calcule la log-probabilité du dataset actuel non découpé, et on la compare à celle du dataset actuel avec le découpage proposé, comme présenté dans la formule suivante :

$$\Delta P(découpage) = \frac{log(P(découpé)) - log(P(non\ découpé))}{log(P(découpé))}, \tag{3.32}$$

avec *log(P(découpé))* défini comme suit :

$$log(P(découpé)) = \sum_{m=1}^{M} \sum_{i=1}^{C_m} log(P(\bigcap_{j=1+\sum_{k=1}^{i-1} c_k^m}^{\sum_{k=1}^{i} c_k^m} B_j = b_j^m)), \tag{3.33}$$

où m désigne la réalisation du réseau, C_m désigne le nombre de champs dans le découpage proposé pour m, c_k^m représente le k^{ime} champ de C_m, et b_j^m désigne la valeur observée du bit à la position j de la réalisation m. *log(P(non découpé))* est défini comme suit :

$$log(P(non\ découpé)) = \sum_{m=1}^{M} log(P(\bigcap_{j=1}^{T_m} B_j = b_j^m)), \tag{3.34}$$

où T_m est le nombre de bits de la réalisation m. Si $\Delta P(découpage)$ est inférieur à un seuil de validation défini par l'utilisateur (hyperparamètre), alors le découpage candidat est accepté, sinon il est rejeté.

3.4 Conclusion

Dans ce chapitre, nous avons commencé par introduire quelques bases des réseaux Bayésiens classiques et dynamiques, à savoir leurs principes fondamentaux, leur représentation graphique, et les procédures d'inférence et d'apprentissage.

Ces réseaux permettent de lier probabilistiquement entre elles de façon graphique toutes les variables présentes dans un système. Il est ensuite possible de calculer à faible coût n'importe quelle distribution de probabilité des variables.

Nous avons ensuite défini l'inférence comme l'opération consistant à propager l'information probabiliste ainsi que les observations effectuées sur les variables à travers le réseau. Dans le cadre de cette thèse, nous ne nous sommes intéressés qu'à l'inférence exacte, et nous avons présenté une procédure possible pour l'effectuer. Celle-ci consiste dans un premier temps à convertir le réseau en un *junction tree*, un nouveau réseau dont les nœuds sont des groupes de variables, et au sein duquel il n'existe qu'un seul chemin reliant deux nœuds. Le *junction tree* est donc une version plus compacte du réseau initial, moins complexe, et sans aucune perte d'informations. Ensuite, nous avons expliqué l'algorithme de *message passing*, dont le principe reprend celui du *Forward-Backward*, et qui est appliqué dans le *junction tree*. Ainsi, les informations sont d'abord toutes rassemblées en un nœud quelconque dans une phase *Collect*, avant d'être redistribuées à tous les noeuds dans une phase *Distribute*.

Suite à cela, nous avons présenté l'inférence de Viterbi, consistant non pas à trouver les distributions de probabilité des variables du réseau, mais l'état global le plus probable du réseau. Afin d'éviter l'explosion exponentielle de la complexité, on utilise la propriété d'indépendance conditionnelle des réseaux Bayésiens pour ne considérer qu'une partie du réseau à la fois, appelée frontière de Markov. Le réseau est ensuite entièrement balayé à l'aide de ces frontières de Markov pour calculer de proche en proche l'état le plus probable du réseau.

Après quoi, nous avons introduit la notion d'apprentissage, consistant à apprendre les distributions de probabilité des variables du réseau. Parmi tous les algorithmes possibles pour effectuer cette tâche, nous avons choisi d'exposer l'algorithme *Expectation-Maximization*. Celui-ci consiste à calculer l'espérance de la vraisemblance des observations, à l'aide de l'inférence, en fonction des paramètres du réseau, puis à maximiser cette espérance de

vraisemblance en modifiant les paramètres du réseau. Ces deux opérations sont répétées itérativement jusqu'à convergence.

Ensuite, nous avons présenté les réseaux Bayésiens dynamiques, qui ont pour particularité dêtre constitués d'une répétition cyclique d'un ensemble de variables (motif), chacune de ces répétitions correspondant à un "instant". L'avantage de ces réseaux réside dans le fait qu'ils peuvent modéliser le fonctionnement d'un système dans le temps avec un nombre fini de variables, quelle que soit la durée considérée.

La procédure d'inférence exacte dans les réseaux Bayésiens dynamiques que nous avons choisi de présenter reprend le même principe que pour les réseaux bayésiens simples, à savoir construire un *junction tree* et appliquer un algorithme de type *Forward-Backward*. Dans les faits, un *junction tree* est construit pour chaque motif, puis ils sont tous reliés les uns aux autres pour former une chaîne. Ensuite, l'algorithme d'interface est utilisé, consistant à transmettre l'information de motif en motif depuis le premier jusqu'au dernier (passe *forward*), puis du dernier au premier (passe *backward*). Les informations sont propagées au sein des motifs à l'aide de l'algorithme de *message passing*.

Après cela, nous avons expliqué que, concernant l'inférence de Viterbi, le principe restait le même à ceci près que chacune des frontières de Markov employées correspondaient généralement à un motif.

Enfin, le principe de l'apprentissage reste identique, à la différence près qu'il faut considérer une dimension supplémentaire, celle du "temps".

Dans la seconde partie du chapitre, nous nous sommes appuyés sur les notions précédentes pour présenter le modèle d'apprentissage de formats de trames proposé dans cette thèse, BaNet3F.

Nous avons d'abord présenté le fonctionnement global du modèle. Pour limiter sa complexité, celui-ci emploie une procédure récursive, dans laquelle, à chaque passe, un découpage candidat en quelques parties est trouvé à l'aide d'un réseau Bayésien, puis est validé ou non en se basant sur la variation de sa vraisemblance entre la version découpée et celle non découpée. La récursion se termine lorsque plus aucun découpage candidat n'est validé.

Le réseau Bayésien employé est considéré comme modèle génératif des trames de la trace à analyser. L'objectif est donc, pour chaque trame, de trouver l'état du réseau le plus probable qui pourrait conduire à sa génération. Il est ensuite possible d'extraire des valeurs des variables le découpage de chaque trame. Pour cela, on ne dispose que de quelques distributions connues et des observations des bits des trames ; il faut donc apprendre les distributions inconnues, puis déterminer à l'aide de l'inférence de Viterbi l'état le plus probable du réseau.

Afin de faciliter une évolution rapide du réseau, nous avons présenté une variante de l'inférence Viterbi, AMBV, incluant un calcul automatique des frontières de Markov ainsi que diverses optimisations. Celles-ci consistent notamment à limiter autant que possible les calculs inutiles où ceux répétés à l'identique de nombreuses fois.

Nous allons ensuite, dans le prochain chapitre, étudier les performances du modèle confronté à diverses traces à analyser.

Chapitre 4

Performances du modèle BaNet3F sur traces générées et traces réelles

Nous avons précédemment présenté le principe du modèle proposé dans cette thèse, BaNet3F, et nous allons maintenant chercher à en évaluer les performances. Pour cela, nous allons confronter notre modèle à deux types de traces différentes : des traces générées, et des traces réelles. On précise que l'objectif de ce chapitre est de se faire une idée du niveau de performance que BaNet3F peut atteindre sur ces deux types de traces, ainsi que de l'influence de ses divers paramètres. Nous chercherons à optimiser les performances du modèle dans le chapitre 5, afin de le comparer à l'état de l'art.

Les traces générées sont constituées d'une suite de champs, chaque champ étant caractérisé par sa position (bit de début et de fin), et ses valeurs possibles. Le générateur tire simplement des valeurs aléatoires pour chacun des champs, en prenant en compte un certain niveau de dépendance entre les valeurs des champs. Les champs générés ne correspondent à aucun protocole, car ici, on cherche à pouvoir librement modifier les caractéristiques de la trace pour observer l'effet sur les performances du modèle.

Les traces réelles sont constituées à partir de trames issues de captures réseaux récupérées sur internet [70, 71, 72]. Toutes les captures sont issues de protocoles utilisant 802.15.4 en couche Liaison de Données. Les traces ont été formées en ne prenant que l'en-tête de chaque trame, et en les rassemblant ensuite par formats compatibles. Étant assez difficile de trouver des traces libres de droits sans les générer soi-même, seules 6 traces réelles ont été formées, mais elles suffisent pour ce que nous cherchons à montrer dans cette thèse.

Les paramètres dont nous allons étudier l'influence sont les suivants :
— Nombre maximal de valeurs différentes de l'observation (NMVDO) (uniquement pour les traces réelles)
— Nombre de valeurs des champs (NVC) (uniquement pour les traces réelles)
— Nombre maximal de champs par trame (NMCT)
— Seuil relatif de validation du découpage (SRVD)
— Nombre maximal d'unités élémentaires considérées simultanément (NMUECS)

Pour évaluer les performances de notre algorithme, nous utiliserons la précision, la couver-

ture, et le score F.

4.1 Paramètres et métriques

4.1.1 Paramètres

Le **Nombre maximal de valeurs différentes de l'observation** est utilisé pour limiter le nombre de valeurs possibles des octets à chaque position des trames. Cela évite une explosion de la complexité du réseau Bayésien. Concrètement, les valeurs les moins probables sont remplacées par une valeur par défaut lors des phases initiales du traitement par le modèle. Ce paramètre peut prendre n'importe quelle valeur entière supérieure ou égale à 1. Plus la valeur est élevée, plus fortes sont censées être les performances du modèle, mais la complexité augmente exponentiellement. On cherche à vérifier si les performances augmentent vraiment avec la valeur de ce paramètre, et trouver un compromis entre performances et temps de calcul.

Le **Nombre de valeurs des champs** est utilisé pour régler le nombre de valeurs possibles des variables ValeurChamps. Ce paramètre peut prendre n'importe quelle valeur entière supérieure ou égale à 2. On s'attend à ce que cette variable présente une valeur optimale maximisant les performances, ce qu'on cherchera à vérifier, et si c'est bien le cas, voir quelle(s) valeur(s) offre(nt) les meilleures performances.

Le **Nombre maximal de champs par trame** est utilisé pour limiter le nombre maximal de champs généré à chaque passe du modèle. Ce paramètre peut prendre n'importe quelle valeur entière supérieure ou égale à 2. Plus la valeur est élevée, plus le modèle peut analyser de traces de grande longueur, et trouver des découpages complexes, mais plus la complexité du modèle augmente. On cherche alors à vérifier si les performances augmentent vraiment continuellement avec la valeur de ce paramètre, ou si une trop grande complexité finit par nuire aux performances, et si c'est le cas, quelle(s) valeur(s) offre(nt) un bon compromis.

Le **Seuil relatif de validation du découpage** est utilisé pour régler la dureté d'acceptation d'un découpage proposé par le réseau. Ce paramètre peut prendre n'importe quelle valeur réelle, mais étant données les valeurs possibles de $\Delta P(\textit{découpage})$ (voir section 3.3.5), il n'a de sens qu'entre 0 et 1. Plus la valeur est élevée, plus le découpage proposé est accepté facilement. On s'attend à ce qu'il existe une valeur optimale en-deça et au-delà de laquelle les performances décroissent. On cherche alors à trouver une valeur offrant un niveau de dureté permettant l'obtention de bonnes performances.

Le **Nombre maximal d'unités élémentaires considérées simultanément** est utilisé pour définir le nombre maximal de bits ou octets que l'on considère comme étant potentiellement dépendant les uns des autres. Ce paramètre peut prendre n'importe quelle valeur entière supérieure ou égale à 2. Plus la valeur est élevée, meilleures sont supposées être les performances du modèle, mais la complexité augmente exponentiellement. On cherche donc à vérifier si les performances augmentent vraiment avec la valeur de ce paramètre, et trouver un compromis entre performances et temps de calcul.

4.1.2 Métriques

Nous utilisons comme métriques la précision, la couverture, et le score F, comme dans le chapitre 2, mais calculées légèrement différemment. La précision et la couverture ne présentent pas d'intérêt pour les traces générées car leurs comportements sont très proches, à un facteur près ; ces métriques ne seront donc pas présentées dans la partie consacrée aux traces générées.

La **précision** quantifie la part de ce qui a été bien détecté par rapport à tout ce qui a été détecté. Elle est définie comme suit :

$$Précision = \frac{Vrais\ positifs}{Vrais\ positifs + Faux\ positifs}. \tag{4.1}$$

Les vrais positifs sont les champs trouvés par le modèle qui sont corrects, c'est à dire dont la longueur et la position (et par conséquent la valeur) correspondent à un champ réel dans la trace. Les faux positifs sont constitués de tous les champs trouvés par le modèle qui ne vérifient pas cette condition.

La **couverture** quantifie la part de ce qui a bien été détecté par rapport à tout ce qu'il y avait à détecter. Elle est définie comme suit :

$$Couverture = \frac{Vrais\ positifs}{Vrais\ positifs + Faux\ négatifs}. \tag{4.2}$$

Les faux négatifs sont les champs présents dans la trace qui n'ont pas été correctement trouvés par le modèle.

Le **score F** est calculé comme une moyenne entre précision et couverture de la façon suivante :

$$Score\ F = \frac{2 \times Précision \times Couverture}{Précision + Couverture}. \tag{4.3}$$

Il s'avère que la précision et la couverture présentent pour la plupart des paramètres les mêmes comportements à un facteur près ; dans ces cas, nous ne présentons que le score F.

Afin de tester les performances de BaNet3F lorsque la dépendance entre champs varie, il nous a paru important de créer une métrique permettant de quantifier cette dépendance. Pour cela, nous avons défini le **ratio de dépendances moyennes** (RDM), une métrique se basant sur la manière dont le modèle utilise la dépendance pour trouver les champs. Elle est définie comme suit :

$$RDM = \prod_{i=1}^{N} \frac{\prod_{j_1=e_{j_1}}^{i-1} \prod_{k_1 \in \{s(i,i),s(j_1,i)\}} max(RD, \frac{1}{RD})^{\frac{1}{K_1 J_1 (n-1)}}}{\prod_{j_2=e_{j_2}}^{i-1} \prod_{k_2 \in \{s(i,i),s(j_2,i)\}} max(RD, \frac{1}{RD})^{\frac{1}{K_2 J_2 (N-n+1)}}}, \tag{4.4}$$

où i représente la position du bit considéré dans une trame du dataset, et j_1 et j_2 représentent le premier bit de la séquence des bits précédents considérée. $e_{j_1} = max(0, i - H), j_1 \in c_1(0), i \in c_0, c_1 \neq c_0$, et $e_{j_2} = max(0, i - H), j_2, i \in !c$, avec

H représentant le nombre maximal de bits considérés simultanément, et c un champ. k_1 et k_2 représentent la valeur du couple de séquences (bit actuel, bits précédents), $s(j_1, i)$ représente les valeurs de la séquence du bit j_1 au bit i inclus. K_1 et K_2 représentent le nombre de couples de valeurs de séquences, J_1 et J_2le nombre de bits précédents, N le nombre de bits moins un dans une trame du dataset, et n le nombre de champs dans la trame.

RD représente le ratio de dépendance, exprimé comme suit :

$$RD = \frac{P(s(i,i)(k))P(s(j,i)(k))}{P(s(i,i)(k) \cap s(j,i)(k))}, \tag{4.5}$$

où $s(j,i)(k)$ représente la k^{me} valeur de la séquence du bit j au bit i inclus, et $P(x)$ représente la probabilité de l'évènement x.

La dépendance entre deux séquences est définie comme le rapport entre le produit des probabilités marginales d'apparition de ces deux séquences et la probabilité d'apparition de ces deux séquences de façon simultanée. Une valeur de 1 traduit une parfaite indépendance, une valeur entre 0 et 1 signifie que la présence de l'une des séquence favorise l'apparition de l'autre, et une valeur entre 1 et l'infini signifie que l'apparition de l'une des séquences défavorise l'apparition de l'autre.

Le numérateur du RDM quantifie la dépendance entre champs sous la forme d'une moyenne géométrique des dépendances entre un bit de début de champ et les bits précédents considérés, pour tous les champs, pour tous les nombres de bits précédents possibles et pour toutes les valeurs du bit et des bits précédents observés dans le dataset.

Le dénominateur du RDM quantifie la dépendance au sein des champs sous la forme d'une moyenne géométrique des dépendances entre un bit et les bits précédents considérés, tous dans un même champ, pour tous les champs, pour tous les nombres de bits précédents possibles et pour toutes les valeurs du bit et des bits précédents observés dans le dataset.

Plus le RDM est fort, plus cela signifie que la dépendance entre champs est importante par rapport à la dépendance au sein des champs, et donc plus il est difficile pour BaNet3F de séparer les champs.

4.2 Performances obtenues avec les traces générées

4.2.1 Générateur de traces

Pour la génération des traces de test des performances du modèle, nous avons opté pour quelque chose de simple à implémenter, réutilisant les réseaux Bayésiens. Ainsi, les traces sont générées comme étant des réalisations d'un réseau Bayésien. Il s'agit donc du procédé inverse de celui d'analyse par le modèle, car les paramètres sont connus, et on génère des observations, tandis que le modèle cherche à apprendre les paramètres à partir des observation existantes.

Ce réseau consiste en une chaîne de variables, représentant chacune un champ, comme illustré sur la Figure 4.1. Chaque variable peut prendre un certain nombre de valeurs,

correspondant chacune à une valeur possible du champ, c'est à dire une séquence de bits.

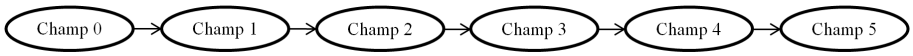


FIGURE 4.1 – Un exemple de réseau utilisé pour générer des trames de 6 champs

On distingue deux types de champs dans une trame : le champ initial (Champ 0 sur la Figure 4.1), ne présentant aucune dépendance, et les autres, présentant une potentielle dépendance au champ qui les précède.

La distribution de probabilité des valeurs possibles du champ initial est calculée comme suit :

$$G(valeur, x) = \begin{cases} (valeur+1)^x \ si \ 0 \leq valeur \leq \dfrac{valeur \ max + 1}{2} - 1 \\ (valeur \ max - valeur + 1)^x \ si \ \dfrac{valeurmax + 1}{2} \leq valeur \leq valeur \ max \end{cases}, \tag{4.6}$$

où *valeur* désigne la valeur décimale de la séquence binaire du champ, et x est le facteur de la distribution intra champ inital, ajustable par l'utilisateur. On normalise ensuite les valeurs calculées afin que les probabilités somment bien à 1. On obtient ainsi une distribution discrète se rapprochant d'une loi normale.

Par exemple, pour un champ de longueur 3 bits et $x = 2$, on obtient la table de probabilités de la Figure 4.2.

Etat000	0.016666...
Etat001	0.066666...
Etat010	0.15
Etat011	0.266666...
Etat100	0.266666...
Etat101	0.15
Etat110	0.066666...
Etat111	0.016666...

FIGURE 4.2 – Un exemple de distribution de champ initial

La distribution de probabilité des valeurs possibles des champs non-initiaux est calculée comme suit :

$$G(valeur, y, z) = \begin{cases} (valeur+1)^{y+z \times VCP} \ si \ 0 \leq valeur \leq \dfrac{valeur \ max + 1}{2} - 1 \\ (valeur \ max - valeur + 1)^{y+z \times VCP} \ si \ \dfrac{valeurmax + 1}{2} \leq valeur \leq valeur \ max \end{cases}, \tag{4.7}$$

où y est le facteur de la distribution intra champs, ajustable par l'utilisateur, et z est le facteur de la distribution inter champs, ajustable par l'utilisateur. C'est ce paramètre qui est utilisé

pour obtenir différentes valeurs de ratio de dépendances moyennes pour toutes les simulations.

VCP (Valeur Champ Précédent), correspondant à la valeur du champ précédent, à laquelle on applique la transformation suivante :

$$G(valeur) = \begin{cases} valeur\ si\ 0 \leq valeur \leq \dfrac{valeur\ max + 1}{2} - 1 \\ valeur\ max - valeur\ si\ \dfrac{valeurmax + 1}{2} \leq valeur \leq valeur\ max \end{cases} . \quad (4.8)$$

On obtient ainsi une distribution discrète se rapprochant d'une loi normale bidimensionnelle.

Par exemple, pour un champ de longueur 3 bits et $y = 5$ et z = -3, on obtient la table de probabilités de la Figure 4.3.

Champ4	Etat_00	Etat_01	Etat_10	Etat_11
Etat_000	0	0	0	0
Etat_001	0	0	0	0
Etat_010	0	0.18	0.18	0
Etat_011	0.5	0.32	0.32	0.5
Etat_100	0.5	0.32	0.32	0.5
Etat_101	0	0.18	0.18	0
Etat_110	0	0	0	0
Etat_111	0	0	0	0

FIGURE 4.3 – Un exemple de distribution de champ non-initial

Afin de ne pas avoir plusieurs datasets avec le même ratio de dépendances moyennes pour un même découpage, on ne retient que ceux ayant des valeurs suffisamment éloignées (on a arbitrairement défini ce seuil à 0.02). En effet, les paramètres x, y et z précédents n'ont qu'un impact indirect sur le ratio de dépendances moyennes, si bien que la valeur obtenue est difficilement prévisible, et parfois très proche pour des x, y et z différents.

Pour limiter l'écart-type des valeurs du ratio de dépendances moyennes pour un même jeu de paramètres de génération, on force à 0 tous les paramètres des tables de distribution de probabilités ne vérifiant pas l'inégalité suivante :

$$paramètre^2 \times nombre\ de\ trames < 10. \quad (4.9)$$

Cette formule signifie que la probabilité de l'apparition simultanée de deux valeurs indépendantes de deux champs distincts doit être suffisante pour que l'espérance de l'apparition de ce couple de valeurs dans la trace générée soit supérieure ou égale à 10.

Enfin, on utilise un hyperparamètre pour limiter le nombre maximal des valeurs d'un champ, afin de maîtriser la complexité du dataset.

4.2.2 Cadre des simulations

Les paramètres des campagnes de simulations visant à évaluer l'influence des paramètres précédemment mentionnés sur les performances de BaNet3F appliqué à l'analyse de traces

Paramètre	**Campagne 1**	**Campagne 2**	**Campagne 3**
Nombre de trames	1000	1000	1000
Facteur de la distribution intra champ inital	3	3	3
Facteur de la distribution intra champs	5	5	5
Facteur de la distribution inter champs	0 ;-1 ;-2 ;-3 ;-4 ;-5	0 ;-1 ;-2 ;-3 ;-4 ;-5	0 ;-1 ;-2 ;-3 ;-4 ;-5
Nombre maximum de valeurs d'un champ	10	10	10
Découpage caché	4,4 3,1,1,1,2 32,32 16,8,8,32 1,2,1,3,1,8,16,32 3,3,2 8,16,32,8 1,4,3,64,2,3,2,1	4,4 3,1,1,1,2 32,32 16,8,8,32 1,2,1,3,1,8,16,32 3,3,2 8,16,32,8 1,4,3,64,2,3,2,1	4,4 3,1,1,1,2 32,32 16,8,8,32 1,2,1,3,1,8,16,32 3,3,2 8,16,32,8 1,4,3,64,2,3,2,1
NMVDO	6	6	6
NVC	2	2	2
NMCT	2 ;3 ;4	4	4
SRVD	0.2	0.12 ;0.14 ;0.16 ;0.18 ;0.2	0.2
NMUECS	4	4	2 ;4 ;8
Nombre de répétitions	10	10	10

TABLE 4.1 – Paramètres des campagnes sur traces générées

générées sont présentés dans la Table 4.1.

On génère des datasets de 1000 trames, car il s'agit d'un compromis acceptable entre le temps de calcul et la fidélité des statistiques aux lois de probabilités utilisées pour la génération.

Les valeurs de facteur de la distribution intra champs et intra champs initial ont été choisies arbitrairement, après avoir simplement constaté qu'elles permettaient de générer des datasets avec une diversité acceptable de RDM.

Les valeurs du facteur de la distribution inter champs permettent de parcourir l'ensemble de l'espace des valeurs du ratio de dépendances moyennes pour un découpage considéré.

On limite le nombre de valeurs possibles de chaque champs à 10 de façon arbitraire.

Concernant les découpages cachés, on en considère 8, aux caractéristiques relativement diverses, afin d'avoir un meilleur aperçu des capacités et limitations du modèle. D'abord, les découpages peuvent être classés en fonction de la taille des champs qu'ils contiennent : 3 découpages ne comportent que des champs d'un octet ou plus, 3 ne comportent que des champs de moins d'un octet, et 2 comportent des champs de n'importe quelle longueur. Ensuite, les découpages peuvent être classés en fonction de leur complexité : 2 ne comportent

que 2 champs, donc ne nécessitent qu'un seul découpage, tandis que les 6 autres comportent entre 3 et 8 champs. Enfin, les découpages peuvent être classés en fonction de s'ils sont trouvables ou non par le modèle. Afin d'améliorer les performances, on restreint les longueurs autorisées à celles trouvées généralement dans les traces binaires : 1,2,3,4,8,16,32,64 bits. Cela a pour conséquence de rendre certains découpages impossibles à trouver lorsque le NMCT est faible. On a donc, parmi les 8 découpages, 5 trouvables avec un NMCT de 2 ou plus, 2 trouvables avec un NMCT de 3 ou plus, et 1 trouvable avec un NMCT de 4 ou plus.

Le nombre maximal de valeurs différentes de l'observation est fixé à 6 pour s'assurer que le nombre de paramètres de la variable octet ne croisse pas au-delà de ce que la mémoire vive de la machine de test peut stocker.

Le nombre de valeurs des champs correspond au nombre d'états de la variable Valeur-Champ, et est fixé à 2 car les valeurs supérieures ne semblent pas améliorer les performances de façon notable, mais augmentent par contre la complexité du réseau du modèle.

Les valeurs du nombre maximal de champs par trame et du seuil relatif de validation du découpage sont celles présentant le plus grand intérêt pour la visualisation de leur impact sur les performances (déterminé a posteriori). Leurs valeurs par défaut ont été fixées arbitrairement à des valeurs permettant d'obtenir des performance correctes.

Les longueurs possibles des champs étant des puissance de 2, et un champ ne pouvant faire plus de bits ou octets, il nous a semblé judicieux d'adopter les mêmes valeurs pour le nombre maximal d'unités élémentaires considérées simultanément.

Enfin, chaque simulation est répétée 10 fois pour pouvoir moyenner les résultats et lisser ainsi les courbes. Un moyennage sur 100 aurait été préférable, mais vus les temps de simulation impliqués, c'était inenvisageable.

Lorsque jugé intéressant pour faire ressortir des tendances ou gommer des points singuliers, nous utiliserons également du lissage par moyennage mobile sur les n points précédents.

Pour chaque paramètre étudié, nous distinguerons huit contextes différents d'évaluation des performances :

- En moyennant les résultats sur tous les découpages
- En moyennant les résultats sur les découpages constitués uniquement de champs de moins d'un octet (3,3,2 3,1,1,1,2 4,4)
- En moyennant les résultats sur les découpages constitués uniquement de champs d'un ou plusieurs octets (8,16,32,8 16,8,8,32 32,32)
- En moyennant les résultats sur les découpages constitués de champs de toutes longueurs (1,4,3,64,2,3,2,1 1,2,1,3,1,8,16,32)
- En moyennant les résultats sur les découpages simples (2 champs) (4,4 32,32)
- En moyennant les résultats sur les découpages complexes (3 champs et plus) (1,4,3,64,2,3,2,1 3,3,2 8,16,32,8 1,2,1,3,1,8,16,32 16,8,8,32 3,1,1,1,2)
- En moyennant les résultats sur les découpages trouvables avec un nombre maximal de champs par trame de 2 ou plus (1,2,1,3,1,8,16,32 16,8,8,32 3,1,1,1,2 32,32 4,4)
- En moyennant les résultats sur les découpages trouvables avec un nombre maximal de

champs par trame de 3 et 4 ou plus (1,4,3,64,2,3,2,1 3,3,2 8,16,32,8)

Le moyennage effectué est assez particulier ; en effet, les paramètres de génération n'ayant qu'une influence indirecte sur le ratio de dépendances moyennes, moyenner les performances de chaque découpage à facteur de la distribution inter champs égal n'aurait pas de sens. On effectue donc un moyennage uniquement lorsque les valeurs de ratio de dépendances moyennes sont suffisamment proches (0.01), sinon, on se contente d'ajouter les points, augmentant ainsi le domaine du graphique.

4.2.3 Influence du nombre maximal de champs par trame sur les performances de BaNet3F

Comme on peut le voir sur le graphique de la Figure 4.4 présentant le score F pour tous les découpages simulés, les performances sont très chaotiques, car le RDM ne quantifie qu'une part de la difficulté de d'analyse de la trace, le reste étant lié au découpage spécifique, et aux particularités du générateur. Cela explique la présence récurrente de creux et de pics. Cependant, on peut tout de même noter la tendance générale à la diminution des performances lorsque le RDM augmente, ainsi que le fait que BaNet3F arrive parfois à trouver le découpage exact des trames des traces analysées dans 100% des cas pour certains découpages et RDM relativement faibles. Lorsque le RDM devient élevé (au delà de 0.3), le modèle ne semble pas pouvoir dépasser les 50% de score F, mais les paramètres en dehors du NMCT n'ayant pas été optimisés, il ne s'agit vraisemblablement pas des meilleures performances obtensibles avec le modèle.

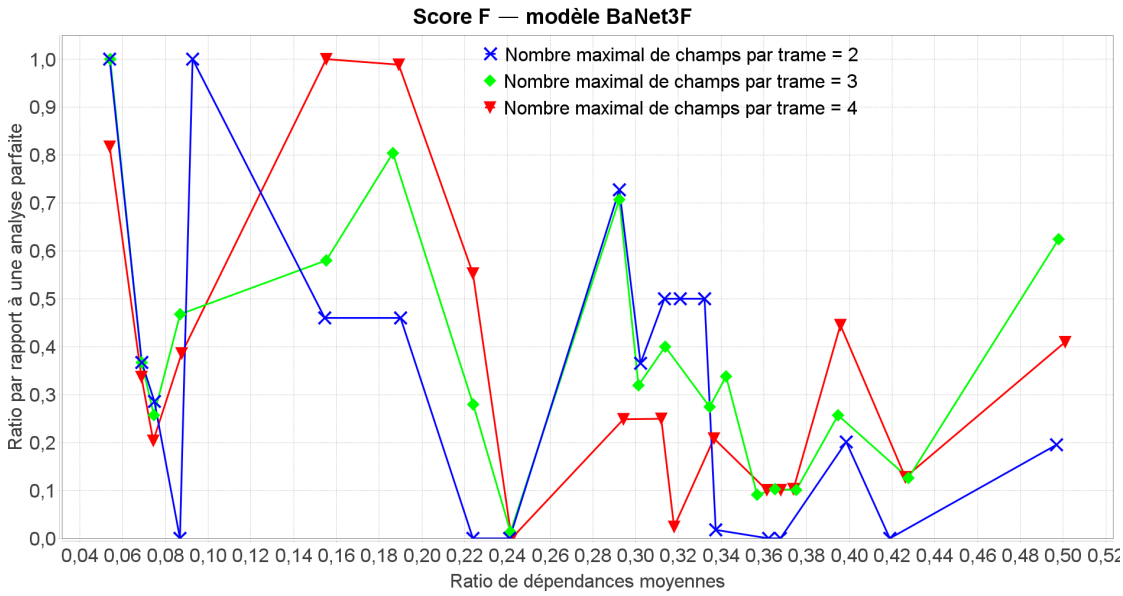


FIGURE 4.4 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur tous les découpages

Pour simplifier la lecture, on a procédé à un lissage sur 5 points, pour obtenir le graphique de la Figure 4.5, où la décroissance globale du score F devient évidente. On peut aussi

voir qu'il n'y a pas vraiment de NMCT optimal, chaque valeur présentant un avantage pour certains intervalles de ratio de dépendances (i. e. certains découpages). Cependant, en poussant le moyennage, on se rend compte que 3 champs par trames présente le meilleur compromis pour les découpages considérés.

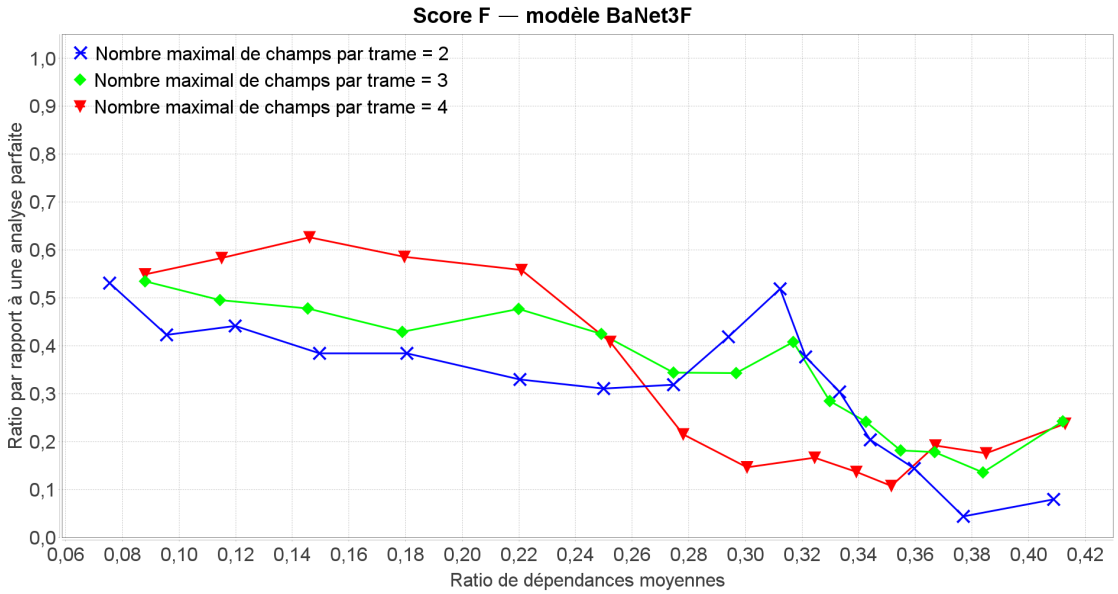


FIGURE 4.5 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur tous les découpages, et lissé sur 5 points

Sur le graphique de la Figure 4.6, présentant le score F pour les découpages dont les champs sont d'une longueur inférieure à l'octet, on constate que la tendance est globablement la même que lorsque tous les découpages sont considérés, à savoir une décroissance du score F avec l'augmentation du RDM et des performances à 100% au maximum, et ne dépassant guère les 50% lorsque le RDM devient élevé. On peut également remarquer que les valeurs de ce dernier ne descendent pas sous les 0.15, ce qui veut dire que les traces ne contenant que des trames courtes d'un octet formées de champs de quelques bits n'ont pas de RDM très faibles.

Le graphique de la Figure 4.7 présente le score F pour des découpages en champs inférieurs à l'octet, auquel est appliqué un lissage sur 5 points. On remarque que la tendance à la décroissance en fonction du RDM est moins marqué, avec une légère remontée sur les fortes valeurs. On constate aussi qu'une valeur de 3 champs maximum par trame est celle donnant les meilleurs résultats.

Le graphique de la Figure 4.8 représente le score F pour les découpages ne contenant que des champs d'un octet ou plus. On peut voir que les valeurs de RDM de ces découpages est compris entre 0.05 et 0.1 ; il s'agit de la partie du graphique de la Figure 4.4 pour les très faibles valeurs du RDM. On peut donc dire qu'ici, la différence entre faibles et fortes performances est uniquement liée aux découpages de la trace à analyser, et non au RDM.

Le graphique de la Figure 4.9 présente le score F pour les découpages contenant des champs

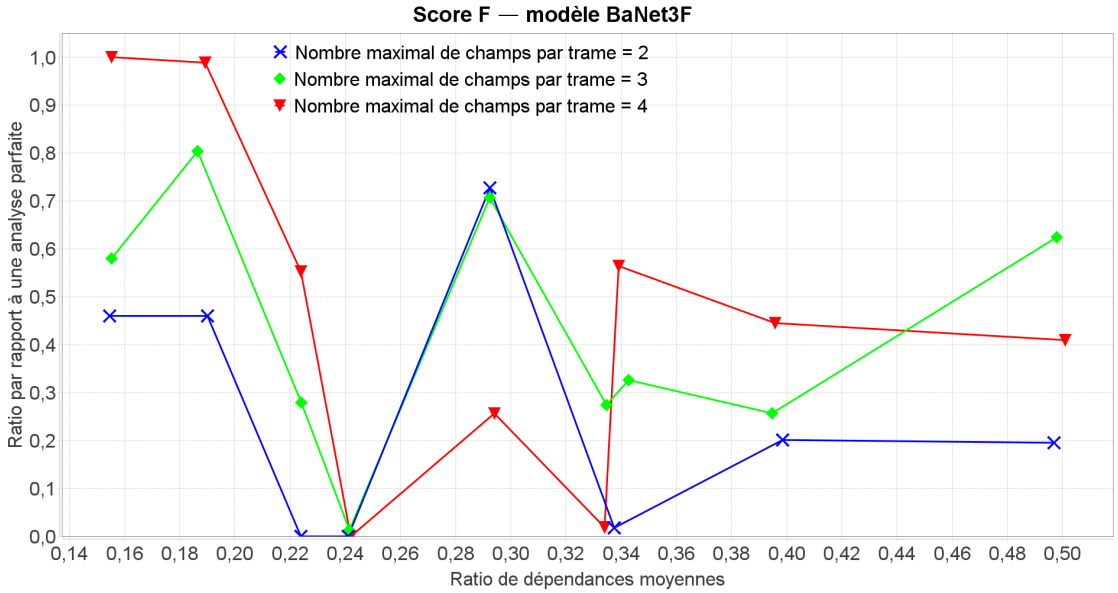


FIGURE 4.6 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les champs de moins d'un octet

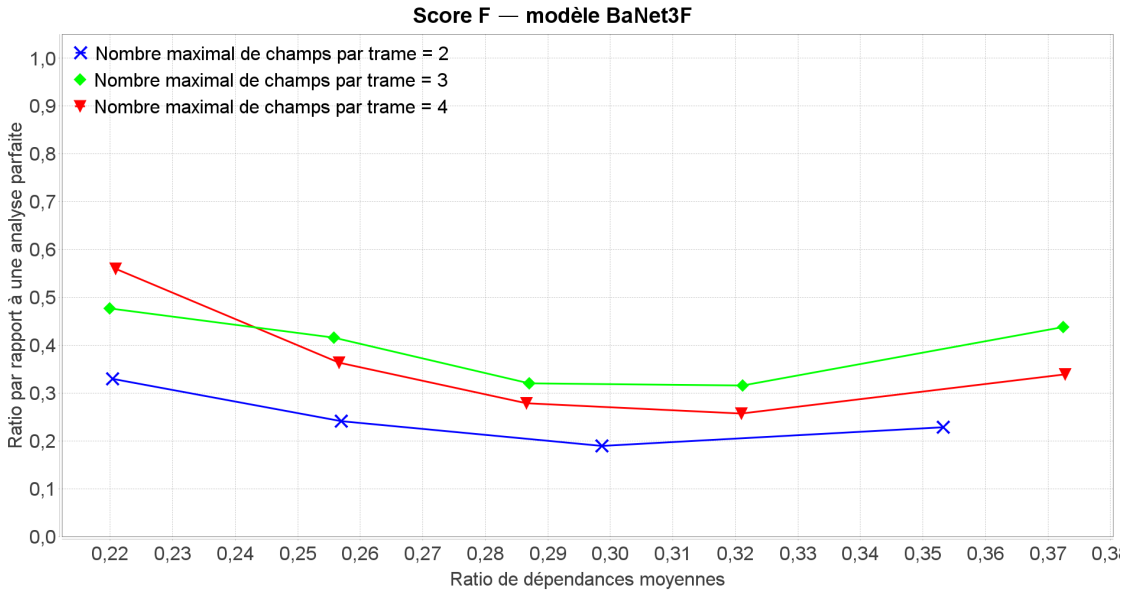


FIGURE 4.7 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les champs de moins d'un octet, et lissé sur 5 points

à la fois de moins d'un octet, et plus d'un octet. On voit clairement que les performances sont moyennes pour les valeurs de RDM les plus faibles, qui correspondent au découpage le plus simple des deux considérés ici pour le modèle, tandis que pour les valeurs les plus élevées, correspondant au découpage le plus complexe, les performances sont mauvaises. Cela s'explique par le fait que le modèle trouve correctement les champs d'un octet ou plus, mais n'arrive pas à découper les octets en champs de quelques bits. Ceci est probablement dû au

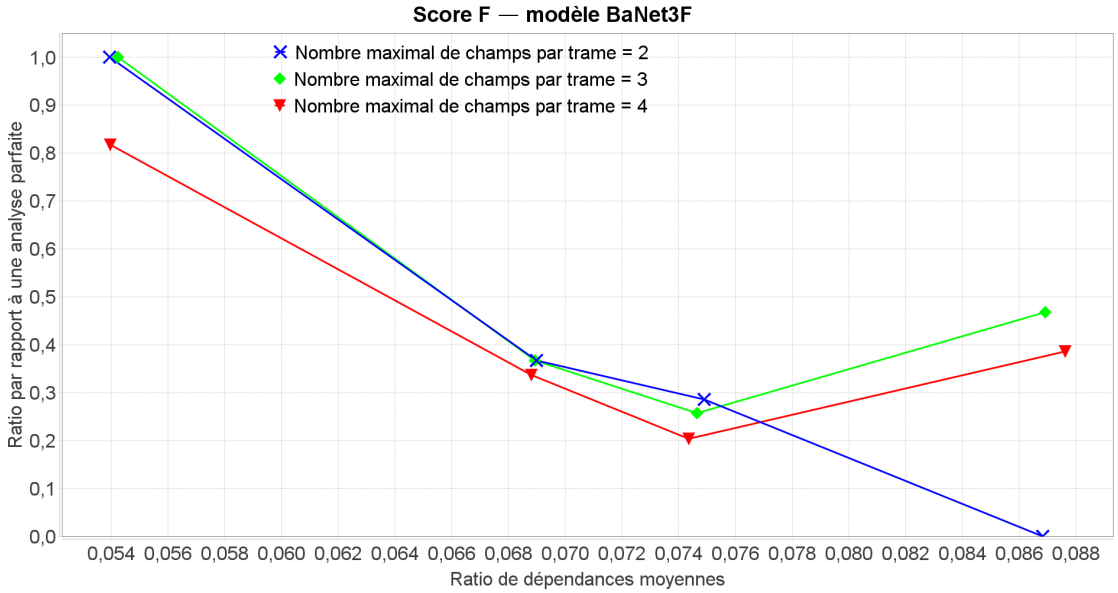


FIGURE 4.8 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les champs d'un octet ou plus

fait que le nombre de valeurs d'un octet constitué de plusieurs champs contient beaucoup de valeurs différentes, et que donc le mécanisme de réduction de la complexité intégré au modèle ne considère que celles les plus présentes, altérant ainsi la trace d'origine.

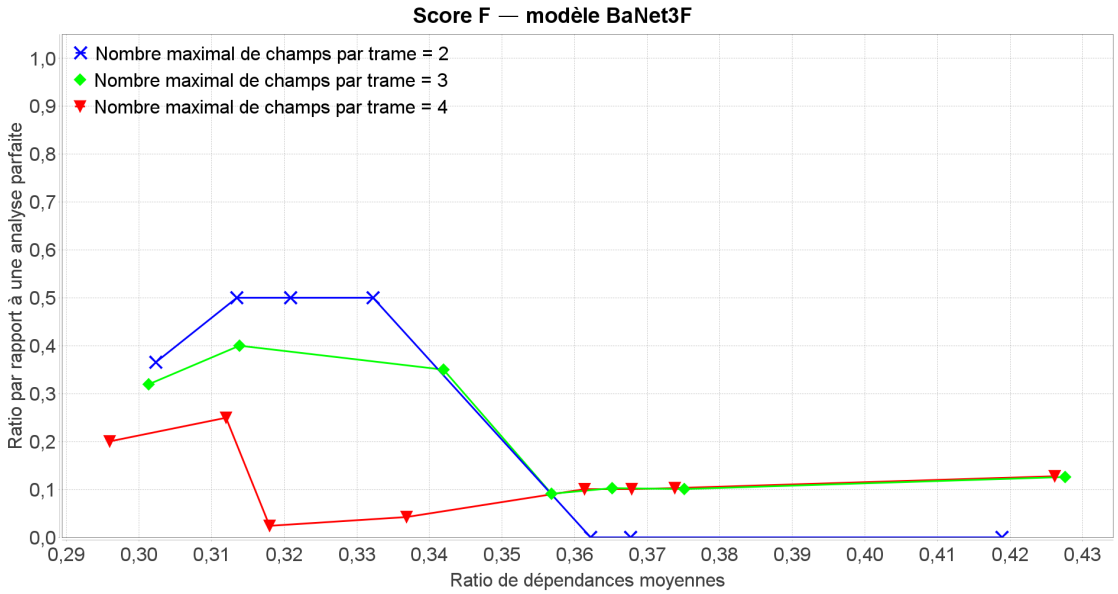


FIGURE 4.9 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages mélangeant des champs de toutes longueurs

Le graphique de la Figure 4.10 présente le score F de BaNet3F pour des découpages

simples ne comportant que 2 champs, et on y constate la même tendance que sur tous les autres graphiques, à savoir la diminution des performances avec l'augmentation du RDM. Cependant, on remarque, de façon surprenante, que plus le NMCT est élevé, meilleures sont les performances. Cela s'explique par le fait que, même si la trace à analyser ne comporte qu'un seul format de trame, le modèle peut décider de la présence de plusieurs formats différents, et va donc les répartir chacun dans des datasets différents pour la passe suivante. Et, en fonction de la manière dont sont répartis les séquences de bits dans ces datasets, la probabilité du découpage évaluée lors de la validation par le modèle change, la faisant passer de la zone d'acceptation vers celle de refus ou vice-versa. On peut donc dire qu'en fonction du NMCT, les formats trouvés à chaque passe ne sont pas les mêmes, donnant ainsi des répartitions différentes.

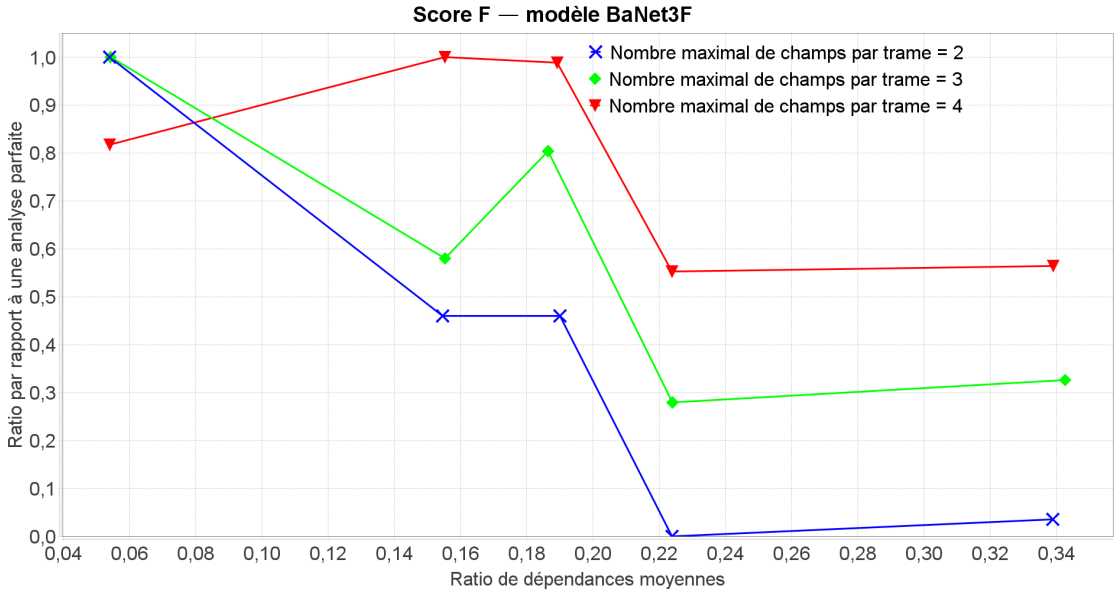


FIGURE 4.10 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages simples

Le graphique de la Figure 4.11 présente le score F de BaNet3F pour des découpages complexes comportant 3 champs ou plus, et, bien que les courbes soient chaotiques, on peut constater que les performances aux faibles valeurs du RDM sont plus basses que pour les découpages simples. cela signifie que BaNet3F est donc généralement moins performant pour ces découpages, ce qui parait logique, car plus le nombre de découpages est important, plus la probabilité de se tromper augmente.

Le graphique de la Figure 4.12 présente la version lissée des courbes du graphique de la Figure 4.11, mettant clairement en évidence que les performances sur des découpages complexes sont nettement inférieures à celles pour des découpages simples, et ce quelle que soit la valeur du RDM.

On peut voir sur le graphique de la Figure 4.13 que pour des découpages trouvables en ne faisant que couper en 2 de façon récurrente les trames, le score F est assez élevé. En effet,

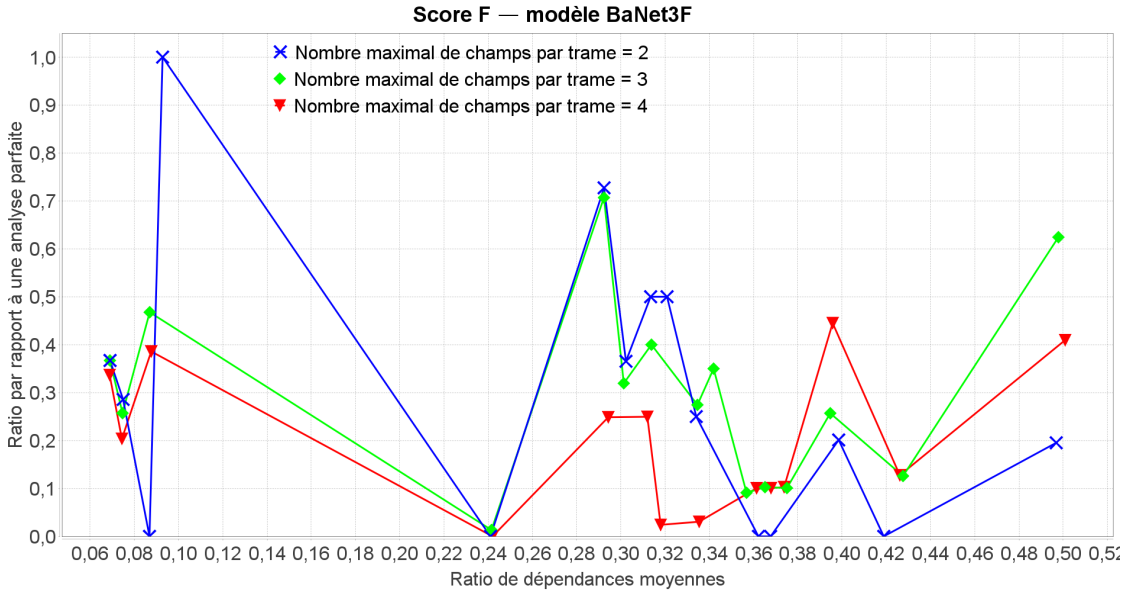


FIGURE 4.11 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages complexes

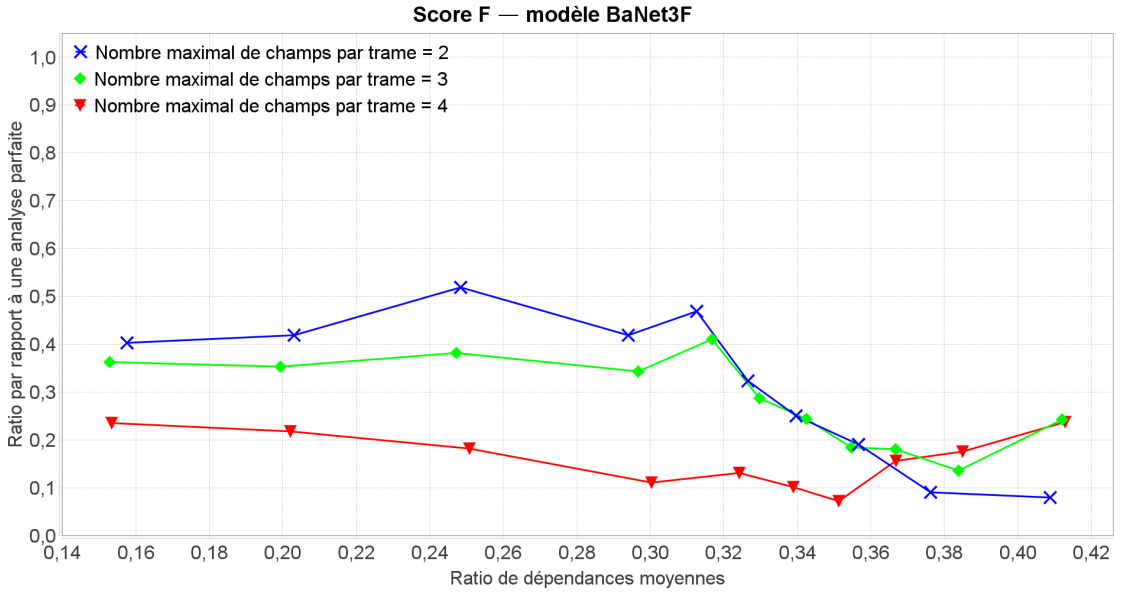


FIGURE 4.12 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages complexes, et lissé sur 5 points

pour les valeurs de RDM faibles, les performances sont autour de 100%, puis elles décroissent jusque légèrement en-dessous de 50% lorsque le RDM augmente.

Le graphique de la Figure 4.14 est la version lissé de la Figure 4.13, mettant clairement en évidence que le score F décroit avec l'augmentation du RDM.

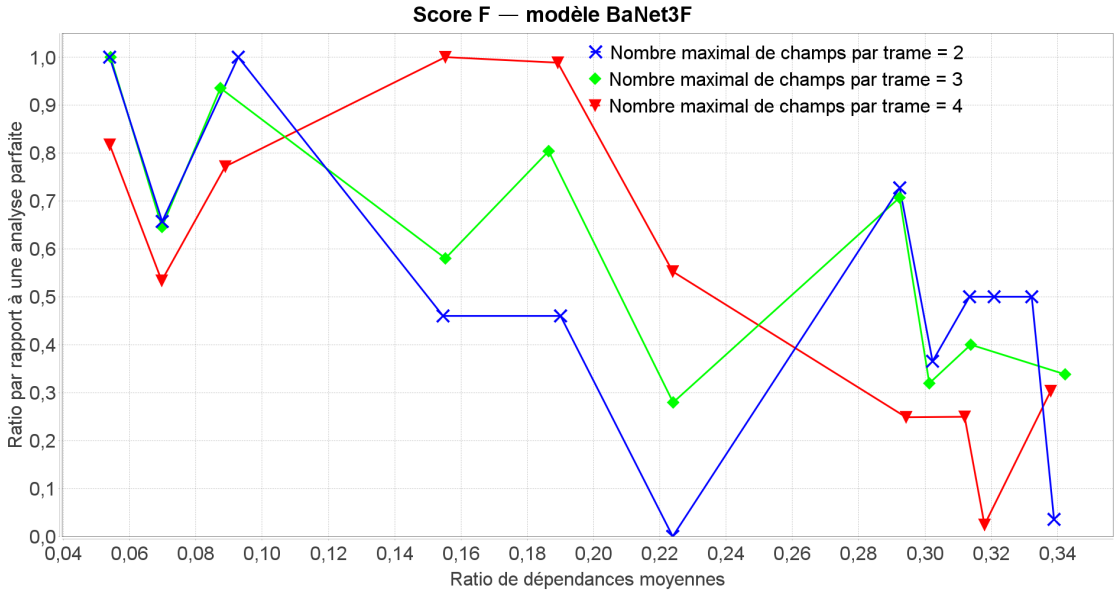


FIGURE 4.13 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages trouvables pour 2 champs maximal par trame ou plus

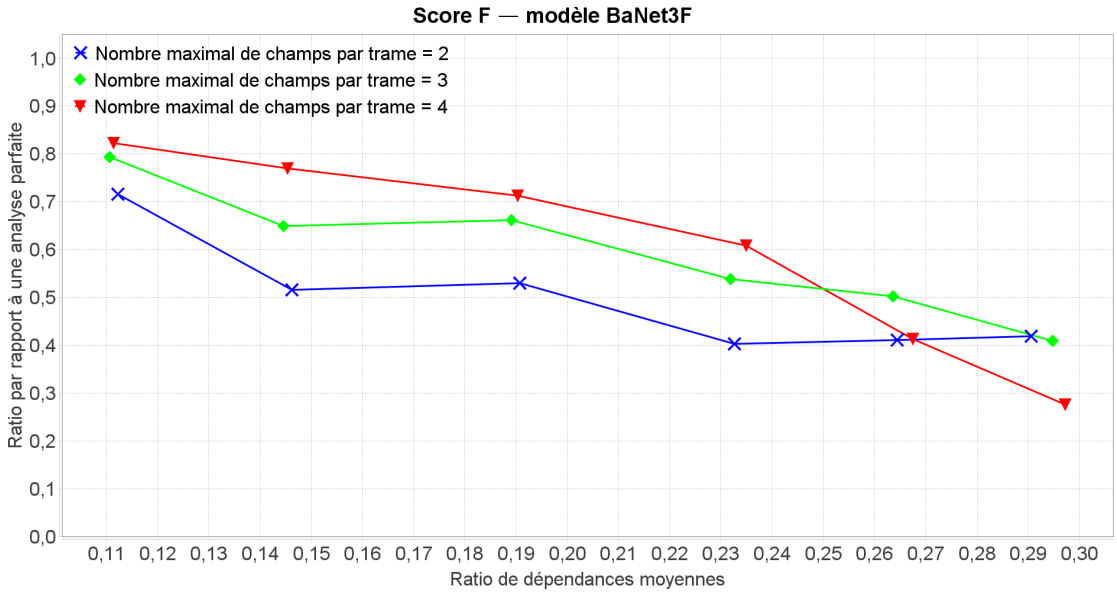


FIGURE 4.14 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages trouvables pour 2 champs maximal par trame ou plus, et lissé sur 5 points

Grâce au graphique de la Figure 4.15 représentant le score F pour les deux découpages impossibles à trouver avec 2 champs par trame maximum, on peut voir que sur ces découpages, qui étaient censés donner de meilleurs résultats pour un NMCT supérieur à 2, le gain en performance est effectivement bien marqué, mais les performances restent malgré tout très faibles globalement. On peut donc dire que BaNet3F n'est pas performant sur les découpages

ne pouvant pas être trouvés en coupant exclusivement en 2 de façon récurrente les trames. On observe également que, contrairement à toutes les courbes précédemment présentées dans cette section, les performances augmentent avec le RDM, ce qui confirme que ce dernier ne suffit pas pour décrire toute la complexité d'analyse des traces.

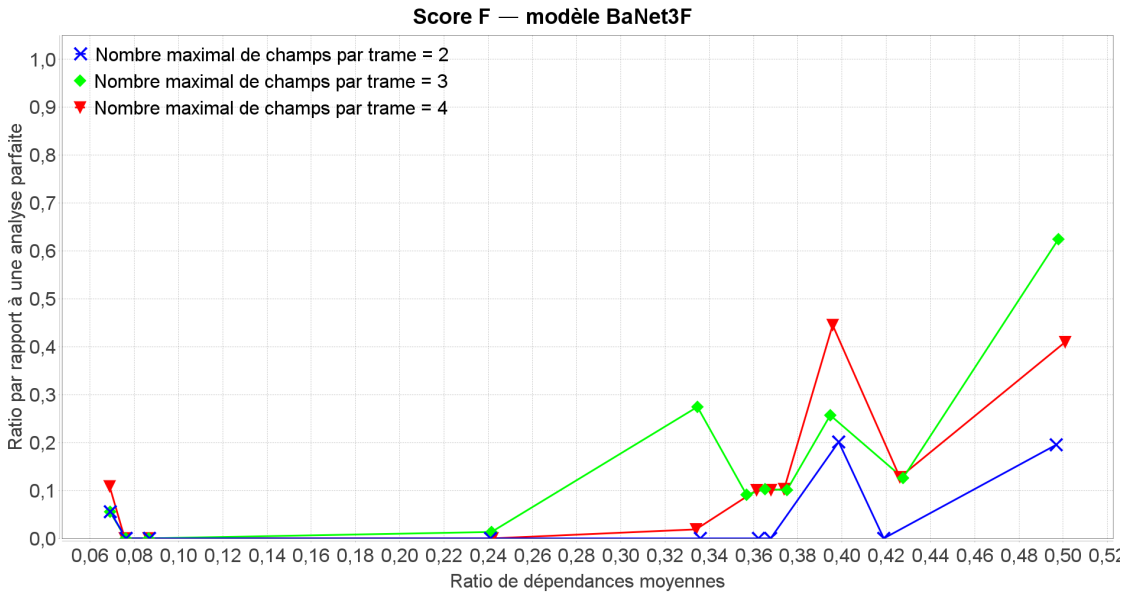


FIGURE 4.15 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages trouvables pour 3 et 4 champs maximal par trame ou plus

4.2.4 Influence du seuil relatif de validation du découpage sur les performances de BaNet3F

Le comportement relatif au RDM étant globalement le même que pour la section précédente, nous en parlerons peu pour cette section et la suivante.

Le graphique de la Figure 4.16 présente la version lissée du score F pour tous les découpages simulés, ce qui permet de montrer que les performances croissent avec le SRVD jusqu'à une valeur de 0.18, et baissent ensuite. On peut donc en déduire que de manière générale, il existe un seuil optimal, potentiellement différent en fonction de la trace à analyser.

Le graphique de la Figure 4.17 présente la version lissée du score F pour les champs de moins d'un octet, et on peut constater que cette fois, le SRVD n'a que peu d'influence sur les performances sous les 0.18. La décroissance globale des performances est plus visible que sur le graphique de la Figure 4.7 car en dessous d'un seuil de 0.18, le score F aux faibles valeurs du RDM est plus important, tandis que la différence aux fortes valeurs est très faible.

Le graphique de la Figure 4.18 présente le score F pour les champs d'un octet ou plus, et on peut y voir qu'une valeur de 0.18 du SRVD ne donne plus les meilleures performances, mais plutôt des valeurs de 0.16 et 0.2. Cela peut s'expliquer par le fait que pour 0.16, certains

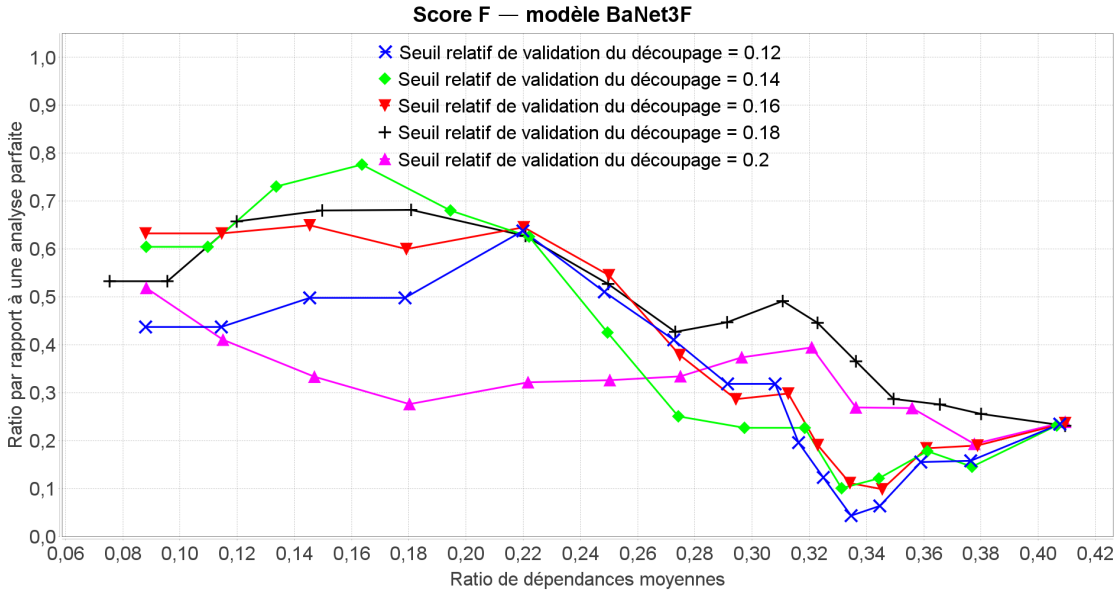


FIGURE 4.16 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur tous les découpages, et lissé sur 5 points

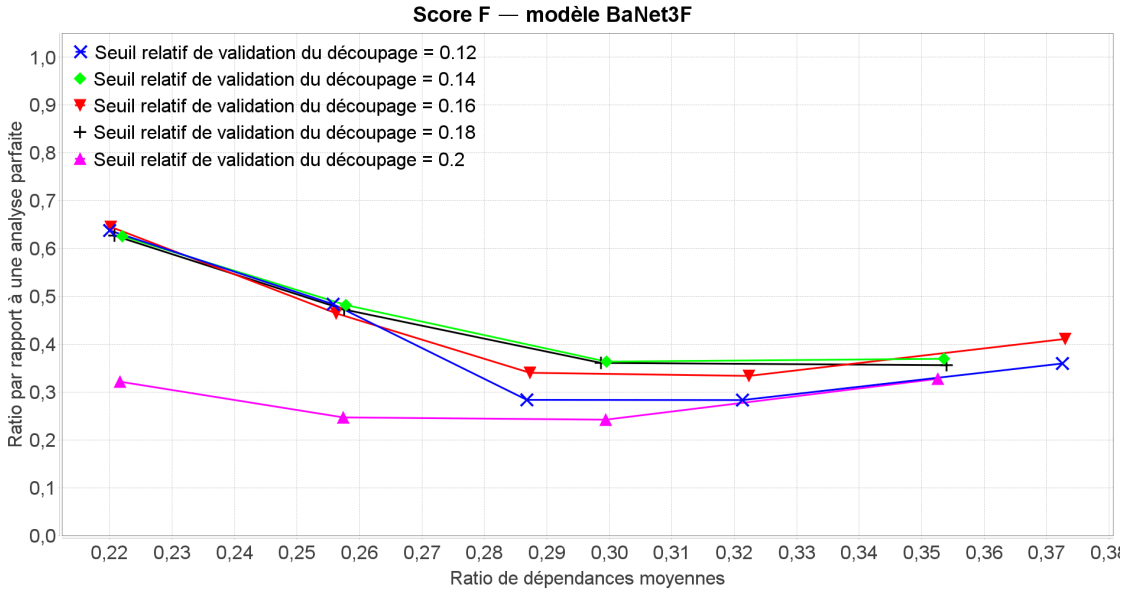


FIGURE 4.17 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les champs de moins d'un octet, et lissé sur 5 points

champs sont trop découpés et les autres juste assez, tandis que pour 0.2, certains champs sont bien découpés, et les autres pas assez. Pour 0.18, certains champs sont trop découpés, et les autres pas assez.

Le graphique de la Figure 4.18 présente le score F lissé pour les champs mélangeant toutes les longueurs. On constate qu'en-deçà d'un SRVD de 0.18, les performances de BaNet3F sont

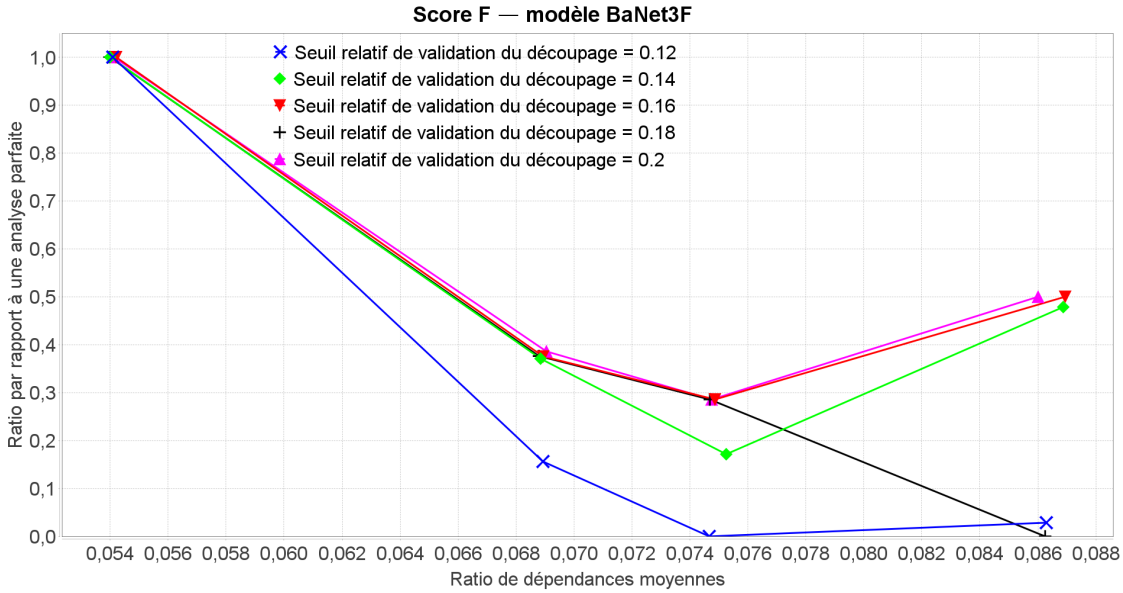


FIGURE 4.18 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les champs d'un octet ou plus

particulièrement faibles, tandis qu'au-delà, elles sont moyennes, mais décroissent jusqu'au niveau des performances en-deça de 0.18 lorsque le RDM augmente.

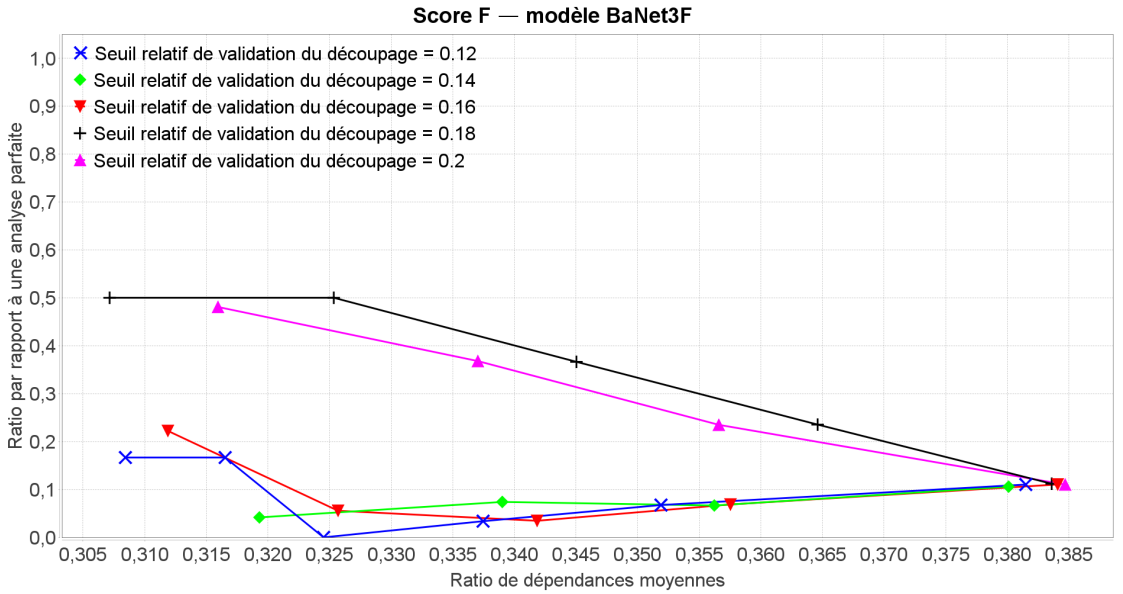


FIGURE 4.19 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages mélangeant des champs de toutes longueurs, et lissé sur 3 points

Le graphique de la Figure 4.20 présente le score F pour les découpages simples, et on peut constater qu'en dehors des valeurs les plus extrêmes du SRVD, les performances varient entre

excellentes et moyennes.

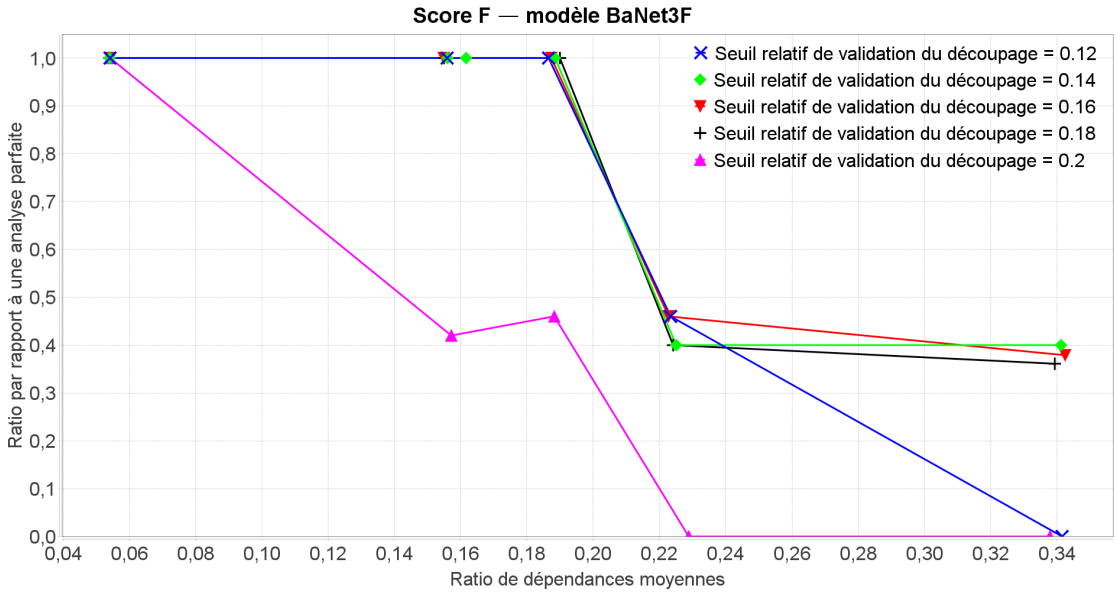


FIGURE 4.20 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages simples

Le graphique de la Figure 4.21 présente le score F lissé pour les découpages complexes, et on peut voir que pour un SRVD de 0.18 les performances sont assez moyennes. Pour des valeurs supérieures à 0.18, les performances sont légèrement inférieures, et pour des valeurs inférieures à 0.18, elles sont même plutôt faibles.

On observe sur le graphique de la Figure 4.22 représentant le score F pour les découpages trouvables avec un NMCT de 2 ou plus, qu'un seuil de validation de 0.18 permet d'obtenir les meilleures performances possibles. Celles-ci sont nettement meilleures que sur le graphique de la Figure 4.14, dépassant les 90% pour les faibles valeurs de RDM, et ne descendant guère sous les 50% pour les valeurs les plus élevées.

On constate sur le graphique de la Figure 4.23, présentant le score F pour les découpages trouvables avec un NMCT de 3 et 4 ou plus, que le SRVD n'a pour ainsi dire pas d'influence, les faibles variations observées pouvant être imputées aux aléas de la génération des traces.

4.2.5 Influence du nombre maximal d'unités élémentaires considérées simultanément sur les performances de BaNet3F

Le graphique de la Figure 4.24 présente le score F lissé pour tous les découpages, et on peut voir que globalement, plus le NMUECS est important, meilleures sont les performances, conformément à nos attentes. On peut également constater que l'écart de performance est assez faible pour les fortes valeurs de RDM, et que l'écart entre 4 unités élémentaires considérées simultanément et 8 est plus faible qu'entre 2 et 4, pour de faibles ou moyennes

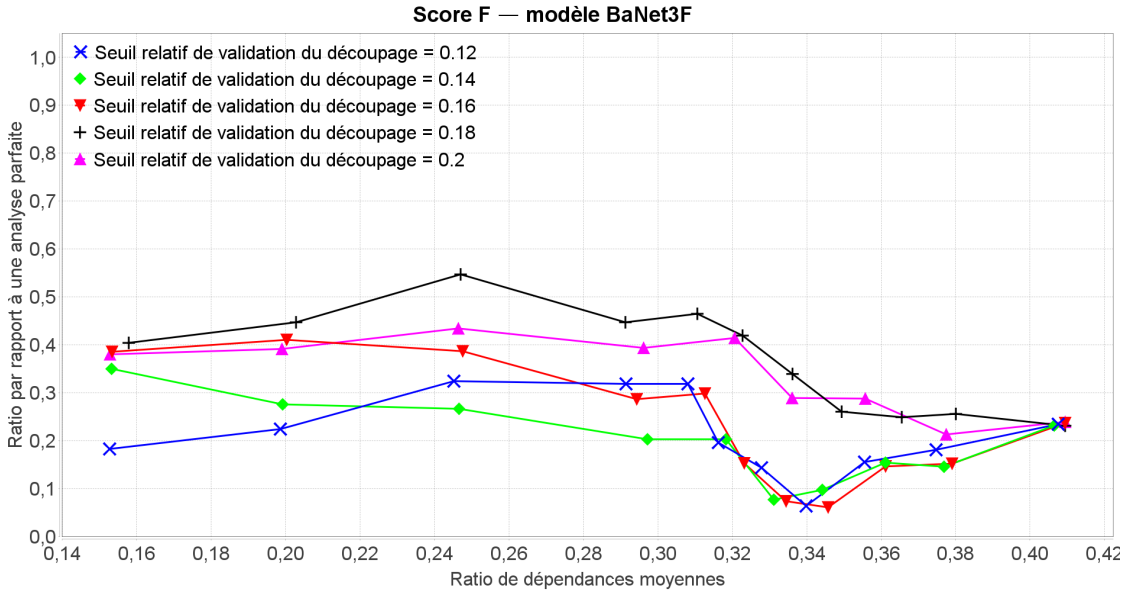


FIGURE 4.21 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages complexes, et lissé sur 5 points

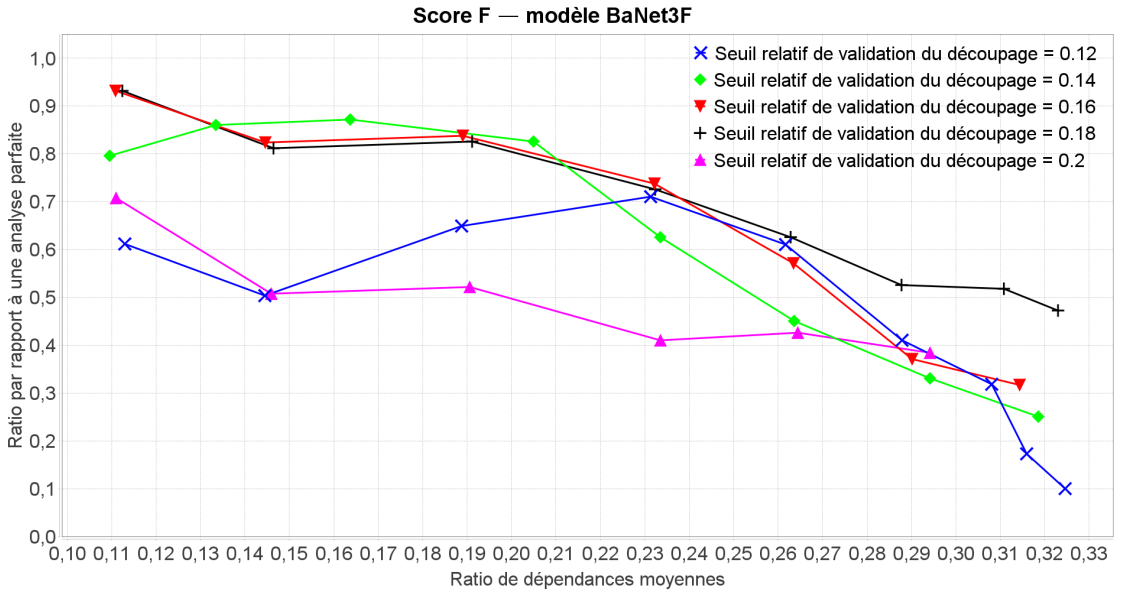


FIGURE 4.22 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages trouvables pour 2 champs maximal par trame ou plus, et lissé sur 5 points

valeurs de RDM. Peut-être est-ce dû au fait qu'aucun des découpages des traces à analyser ne contient de champs de 8 unités élémentaires, mais toujours est-il qu'étant donné l'énorme surcoût en puissance de calcul de 8 unités élémentaires par rapport à 4, cette dernière valeur constitue un bon compromis, comme nous l'espérions.

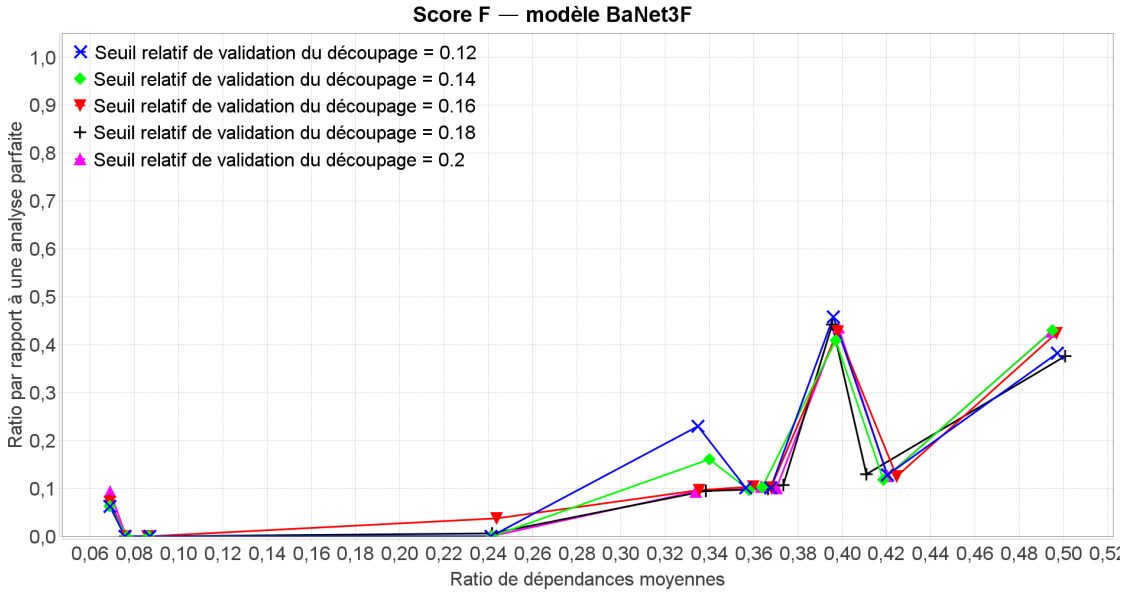


FIGURE 4.23 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages trouvables pour 3 et 4 champs maximal par trame ou plus

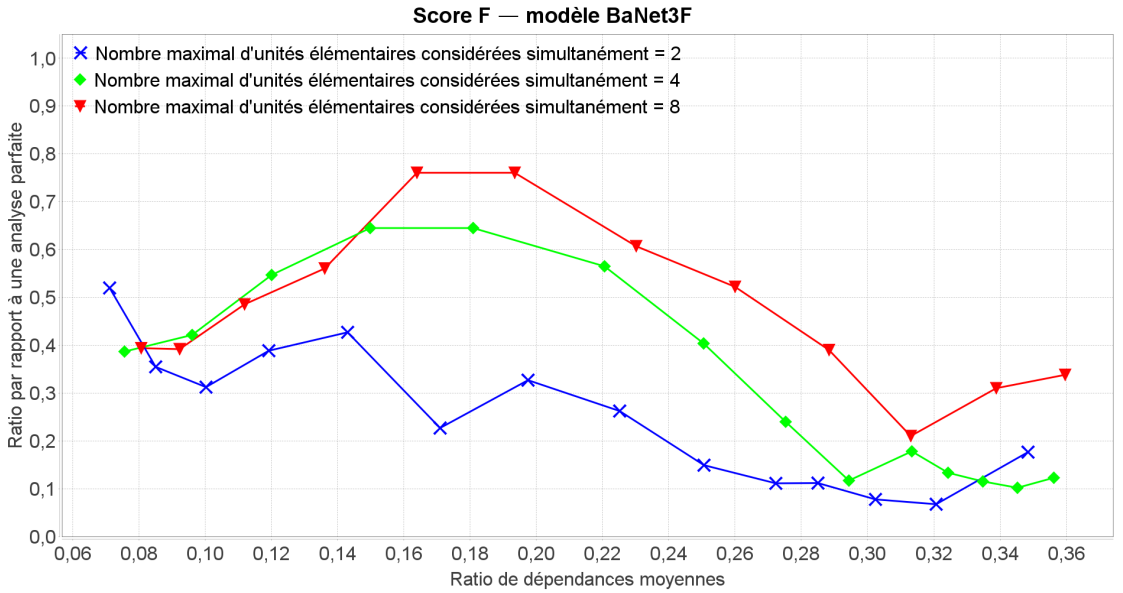


FIGURE 4.24 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur tous les découpages, et lissé sur 5 points

Le graphique de la Figure 4.25 présente le score F lissé pour les découpages ne contenant que des champs de moins d'un octet, et on peut voir que les performances croissent également avec le NMUECS. Cependant, l'écart de performance entre 2 et 4 pour les faibles et moyennes valeurs du RDM est beaucoup plus prononcé que sur le graphique de la Figure 4.24, tandis que pour les fortes valeurs, l'écart s'annule complètement.

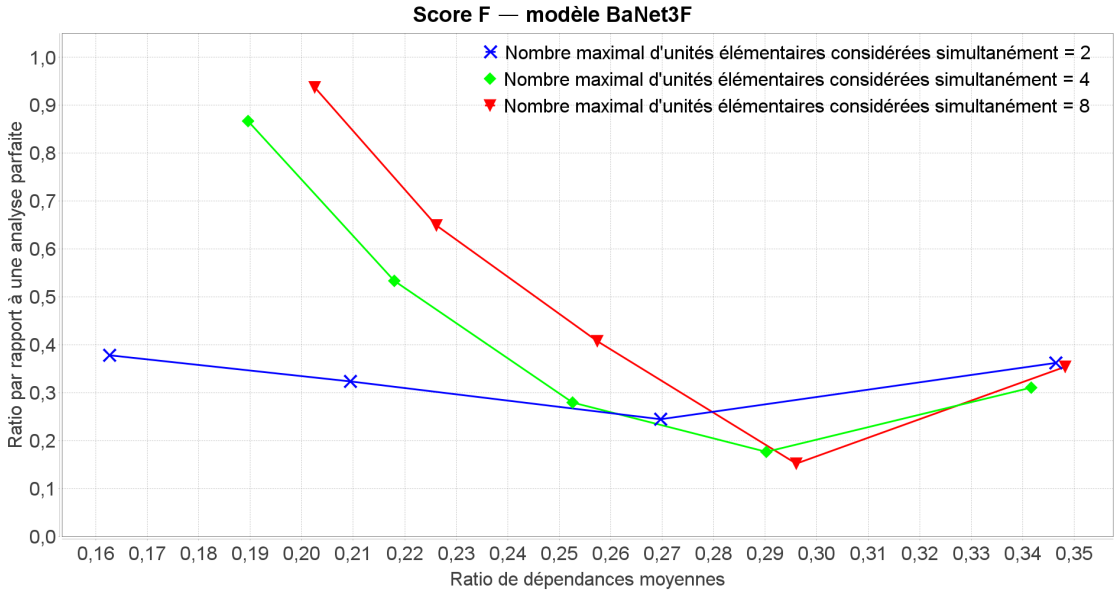


FIGURE 4.25 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur les champs de moins d'un octet, et lissé sur 3 points

Le graphique de la Figure 4.26 présente le score F pour les découpages ne contenant que des champs d'un octet ou plus, et on peut constater que les performances continuent à croitre lorsque le NMUECS augmente, mais que cette fois l'écart entre les différentes valeurs est moins important pour les faibles valeurs du RDM que ce qui a été observé pour les précédents graphiques.

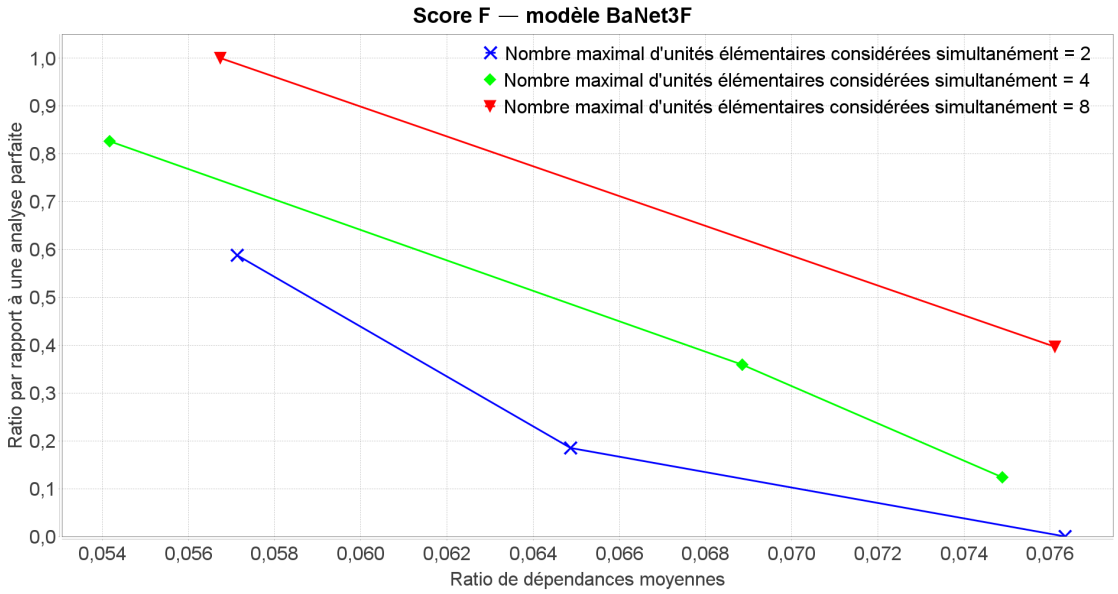


FIGURE 4.26 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur les champs d'un octet ou plus

Le graphique de performances de BaNet3F appliqué aux découpages mélangeant des champs de toutes longueurs n'étant pas suffisamment exploitable, nous ne le présenterons pas.

Le graphique de la Figure 4.27 présente le score F lissé pour les découpages simples, et on peut observer que pour un NMUECS de 4 ou 8, on obtient d'excellentes performances, tandis que pour 2, les performances sont moyennes.

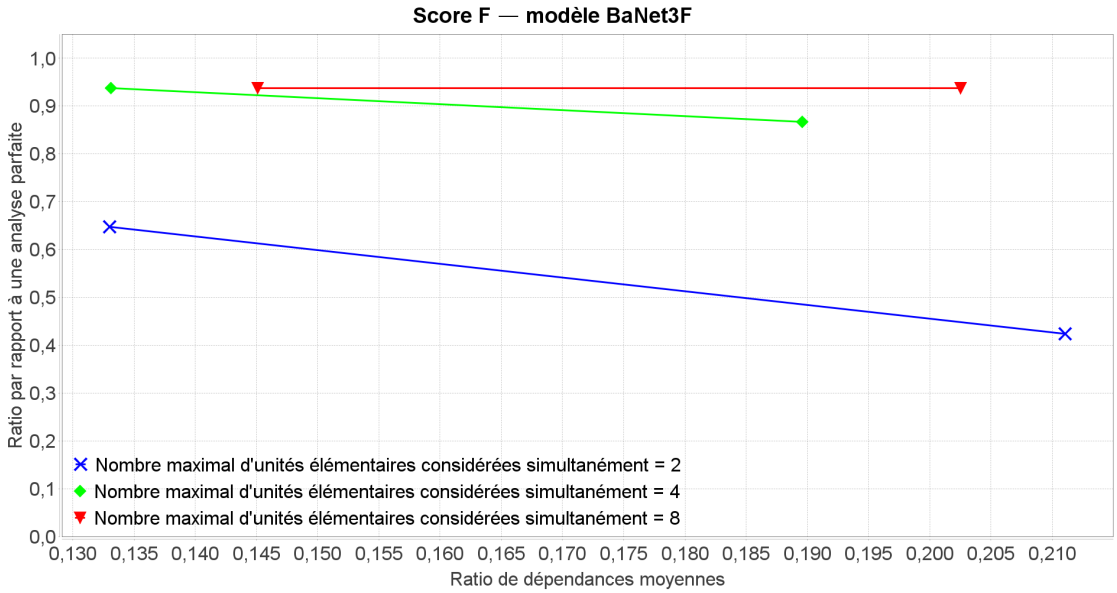


FIGURE 4.27 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur les découpages simples, et lissé sur 3 points

Le graphique de la Figure 4.28 présente le score F lissé pour les découpages complexes, et on peut constater que les performances sont nettement inférieures à celles des découpages simples, comme nous l'avons vu lors de l'analyse de l'influence des autres paramètres, mais cette fois-ci l'écart est encore plus prononcé : en effet, les meilleures performances restent inférieures aux moins bonnes performances du graphique de la Figure 4.27.

Les courbes du graphique de la Figure 4.29, présentant le score F lissé pour les découpages trouvables avec 2 champs maximal par trame ou plus, offrent un comportement similaire aux autres courbes de découpages trouvables avec une valeur du NMCT de 2 ou plus, à savoir de bonnes voire excellentes performances pour les faibles RDM, et des performances faibles à moyennes pour les fortes valeurs.

On constate sur le graphique de la Figure 4.30, présentant le score F lissé pour les découpages trouvables avec 3 et 4 champs maximal par trame ou plus, que le NMUECS n'a pour ainsi dire aucune influence sur les performances, qui restent faibles ou très faibles.

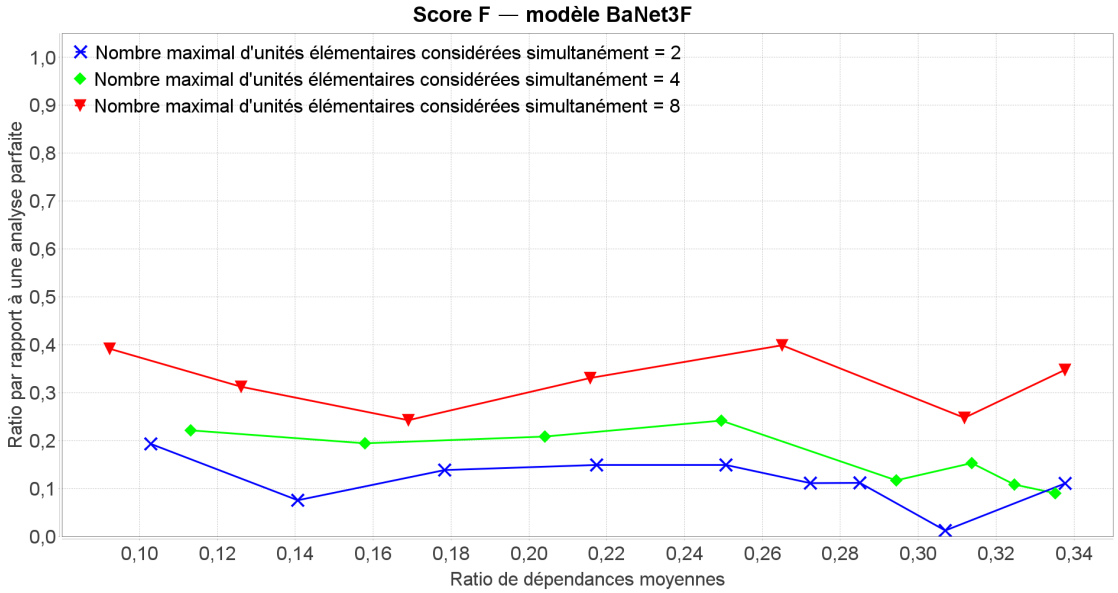


FIGURE 4.28 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur les découpages complexes, et lissé sur 5 points

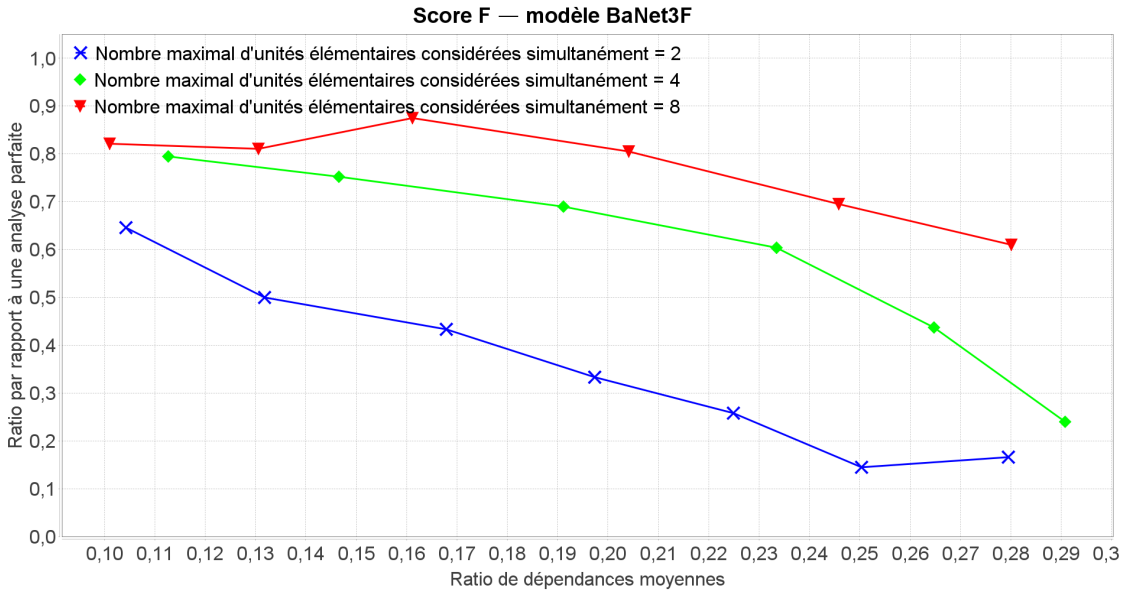


FIGURE 4.29 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur les découpages trouvables pour 2 champs maximal par trame ou plus, et lissé sur 5 points

4.2.6 Bilan

On peut constater que, globalement, les performances de l'analyse des traces générées par BaNet3F baissent lorsque le ratio de dépendances moyennes augmente, en dehors de quelques cas particuliers de découpages spécifiques, pour lesquels cette métrique n'a pas de lien avec

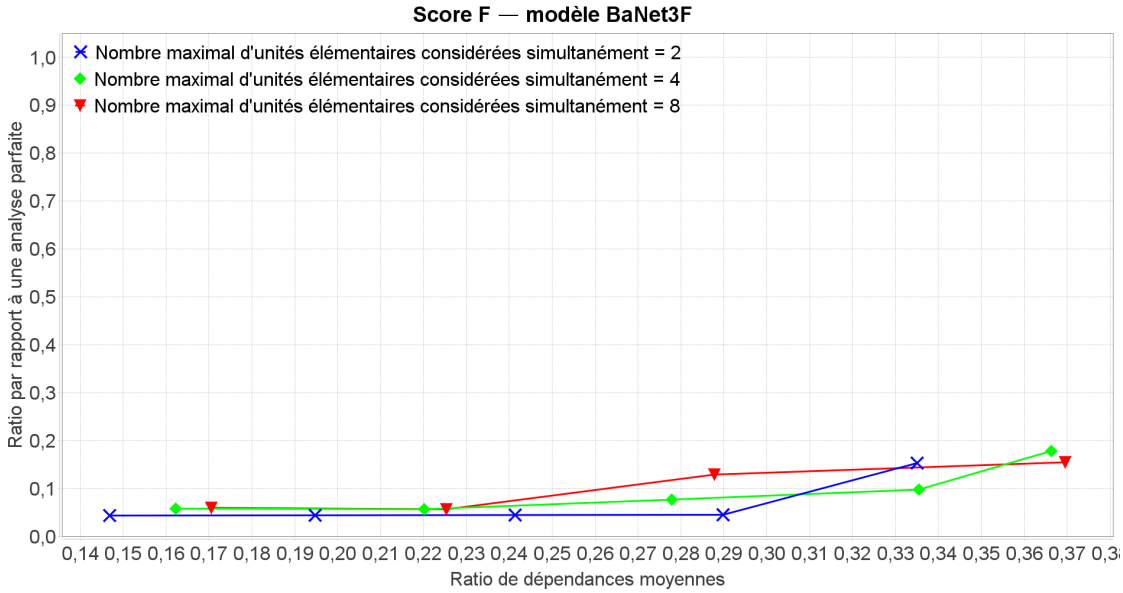


FIGURE 4.30 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur les découpages trouvables pour 3 et 4 champs maximal par trame ou plus, et lissé sur 5 points

les performances. Ce constat correspond bien à nos attentes étant donné le principe de calcul de la métrique.

Concernant l'influence du nombre maximal de champs par trame, il n'y a pas de tendance claire, mais une valeur de 3 semble constituer le meilleur compromis. Cela ne coïncide pas complètement avec ce que nous pensions a priori, mais ce résultat n'est pas non plus une surprise totale, car nous l'avions aussi envisagé.

Pour ce qui est du seuil relatif de validation du découpage, en dehors d'un seul cas particulier (Figure 4.18), tous les autres graphiques indiquent clairement que 0.18 est la valeur donnant les meilleures performances.

Enfin, un nombre maximal d'unités élémentaires considérées simultanément de 8 permet d'obtenir les meilleures performances, comme nous le supposions. Une valeur de 4 semble également être un bon compromis entre performances et complexité du réseau du modèle.

Concernant les découpages de moins d'un octet, les performances sont généralement moyennes pour les faibles valeurs du ratio de dépendances moyennes (excellentes pour un nombre maximal d'unités élémentaires considérées simultanément de 8), et faibles pour les valeurs du ratio de dépendances moyennes.

Pour les découpages d'un octet ou plus, les performances ne présentent pas de corrélation avec le ratio de dépendances moyennes, et les performances varient de parfaites (100%) à nulles (0%) suivant la valeur du facteur de la distribution inter-champs.

Pour ce qui est des découpages en champs de toutes longueurs, les performances varient de moyennes à très faibles lorsque le ratio de dépendances moyennes augmente.

Concernant les découpages simples, les performances sont systématiquement très élevées, voire parfaites lorsque le ratio de dépendances moyennes est faible, et elles diminuent ensuite jusqu'à des valeurs moyennes, faibles, ou même nulles lorsque le ratio de dépendances moyennes devient plus important.

Pour les découpages complexes, les performances semblent n'être que peu, voire pas corrélées au ratio de dépendances moyennes, et varient de moyennes à très faibles en fonction du jeu de paramètres employé.

Pour ce qui est des découpages trouvables pour une valeur du nombre maximal de champs par trame de 2 ou plus, en fonction du jeu de paramètre utilisé, les performances varient de bonnes à faible, ou d'excellentes à moyennes, lorsque le ratio de dépendances moyennes augmente.

Enfin, concernant les découpages trouvables pour une valeur du nombre maximal de champs par trame de 3 et 4 ou plus, les performances ne présentent pas de corrélation avec le ratio de dépendances moyennes, et elles varient de moyennes à nulles, mais sont dans l'ensemble très faibles.

4.3 Performances obtenues avec les traces réelles

4.3.1 Caractéristiques de la trace et cadre des simulations

Afin de tester les performances de BaNet3F, nous avons créé 6 traces à partir de trames réelles issues de protocoles utilisant 802.15.4 en couche Liaison de Données [70, 71, 72].

La première est issue d'un mélange de 6 traces réelles dont seul le header d'un des formats de trames présents a été récupéré. Les 5 autres sont chacune issue d'une trace différente dont seul le header d'un format de trame a été retenu. Les 6 traces réelles constituées et leurs caractéristiques principales sont présentées dans la Table 4.2.

Nom	**Découpage caché**	**Nombre de trames**
Trace 1	3,1,1,1,1,1,1,1,2,2,2,8,16,16,16	92
Trace 2	3,1,1,1,1,1,1,1,2,2,2,8,16,16,16	81
Trace 3	3,1,1,1,1,1,1,1,2,2,2,8,16,16,16	1548
Trace 4	3,1,1,1,1,1,1,1,2,2,2,8	140
Trace 5	3,1,1,1,1,1,1,1,2,2,2,8	478
Trace 6	3,1,1,1,1,1,1,1,2,2,2,8	1596

Table 4.2 – Caractéristiques des traces réelles

Les paramètres des campagnes de simulations visant à évaluer l'influence des paramètres mentionnés dans la section 4.1.1 sur des traces réelles sont présentés dans les Tableaux 4.3 et 4.4.

Paramètre	**Campagne 1**	**Campagne 2**
Trace	Trace 1 Trace 2 Trace 3 Trace 4 Trace 5 Trace 6	Trace 1 Trace 2 Trace 3 Trace 4 Trace 5 Trace 6
NMVDO	4 ;5 ;6 ;7	6
NVC	2	3 ;4 ;5
NMCT	2	2
SRVD	0.4	0.4
NMUECS	4	4

TABLE 4.3 – Paramètres des campagnes sur traces réelles

Paramètre	**Campagne 3**	**Campagne 4**	**Campagne 5**
Trace	Trace 1 Trace 2 Trace 3 Trace 4 Trace 5 Trace 6	Trace 1 Trace 2 Trace 3 Trace 4 Trace 5 Trace 6	Trace 1 Trace 2 Trace 3 Trace 4 Trace 5 Trace 6
NMVDO	6	6	6
NVC	2	2	2
NMCT	2 ;3 ;4 ;5	2	2
SRVD	0.4	0.2 ;0.3 ;0.4 ;0.5 ;0.6	0.4
NMUECS	4	4	2 ;4 ;8

TABLE 4.4 – Paramètres des campagnes sur traces réelles

Les valeurs du nombre maximal de valeurs différentes de l'observation, nombre de valeurs des champs, nombre maximal de champs par trame, et seuil relatif de validation du découpage sont celles présentant le plus grand intérêt pour la visualisation de leur impact sur les performances (déterminé a posteriori). Leurs valeurs par défaut ont été choisies afin de permettre d'obtenir des performance correctes.

Les longueurs possibles des champs étant des puissance de 2, et un champ ne pouvant faire plus de 8 bits ou octets, il nous a semblé judicieux d'adopter les mêmes valeurs pour le nombre maximal d'unités élémentaires considérées simultanément.

Les simulations ne sont pas répétées car l'analyse du modèle est déterministe, ce qui signifie qu'étant donné que les traces ne sont pas générées aléatoirement, le résultat de l'analyse sera toujours le même.

Lorsque jugé intéressant pour faire ressortir des tendances ou gommer des points singuliers, nous utiliserons également du lissage par moyennage mobile sur les n points précédents.

Pour chaque paramètre étudié, nous distinguerons trois contextes différents d'évaluation des performances :

— En moyennant les résultats sur tous les découpages

— En moyennant les résultats sur les découpages longs (3,1,1,1,1,1,1,1,2,2,2,8,16,16,16)
— En moyennant les résultats sur les découpages courts (3,1,1,1,1,1,1,1,2,2,2,8)

4.3.2 Influence du nombre maximal de valeurs différentes de l'observation sur les performances de BaNet3F

Le graphique de la Figure 4.31 présente le score F lissé pour tous les découpages, et on peut constater que les performances croissent légèrement avec le RDM, ce qui est à l'opposé de ce que l'on s'attendait. Cependant, cette tendance étant très faible, on peut considérer les performances comme constantes, ce qui peut être imputé aux spécificités des traces considérées. Par ailleurs, la valeur du NMVDO semble faire baisser les performances lorsqu'elle est trop faible, mais à partir de 5, son influence semble être à peu près inexistante. On remarque également que les performances sont globalement nettement inférieures à celles obtenues avec les traces générées. Nous ne présentons pas la courbe de performance liée à un NMVDO de 8 car elle s'est avérée peu exploitable.

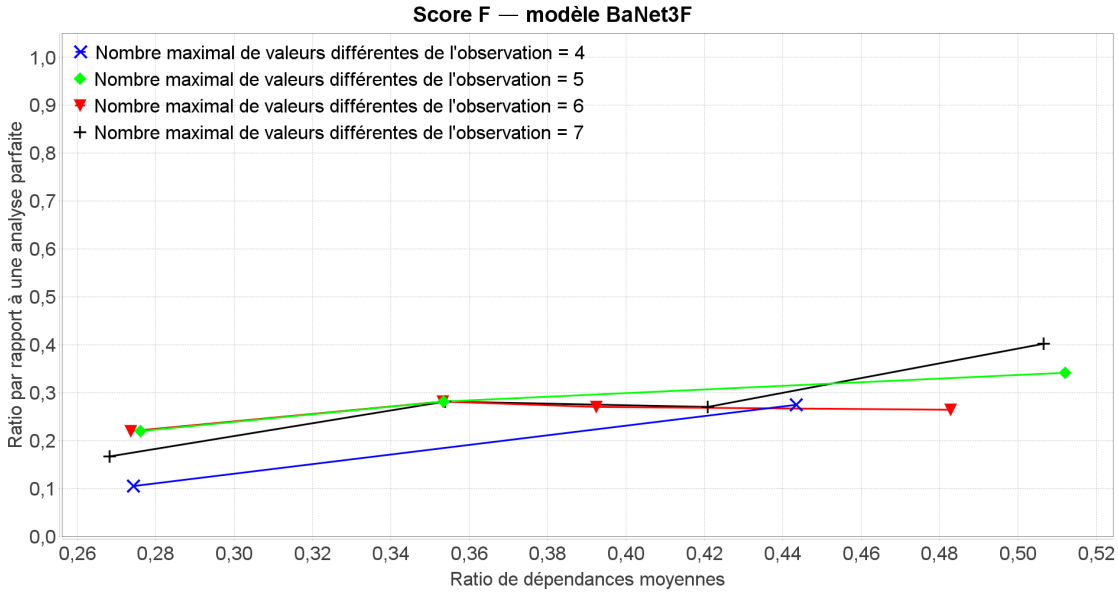


FIGURE 4.31 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMVDO, moyenné sur tous les découpages, et lissé sur 3 points

On peut voir sur le graphique de la Figure 4.32, représentant le score F pour les découpages longs, qu'en dehors du cas de la valeur la plus faible du NMVDO, les performances diminuent légèrement avec l'augmentation du RDM. En observant les performances lorsque le NMVDO croit, on se rend compte qu'elles présentent un maximum pour une valeur de 5.

Le graphique de performances de BaNet3F appliqué aux découpages courts n'étant pas suffisamment exploitable, nous ne le présenterons pas.

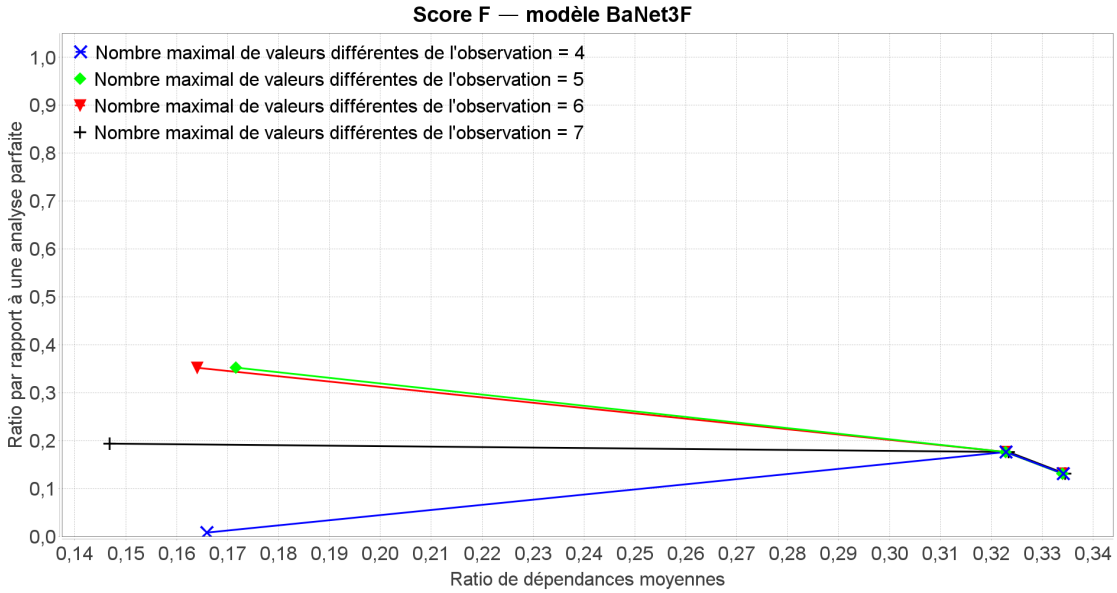


FIGURE 4.32 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMVDO, moyenné sur les découpages longs

4.3.3 Influence du nombre de valeurs des champs sur les performances de BaNet3F

On constate, sur le graphique de la Figure 4.33 présentant le score F lissé pour tous les découpages, que les performances sont relativement constantes quel que soit le RDM, comme sur le graphique de la Figure 4.31. On peut également voir que les performances de BaNet3F augmentent lorsque le NVC augmente ; cependant, au-delà de 5, les performances plafonnent.

Sur le graphique de la Figure 4.34, présentant le score F pour les découpages longs, on constate que les performances restent globalement constantes pour un NVC de 3 ou 4, tandis qu'elles augmentent pour une valeur de 5. L'explication la plus plausible est qu'une fois encore, les performances sont plus liées aux traces spécifiques analysées qu'à la valeur de leur RDM. On remarque également que les performances sont moins bonnes pour deux valeurs des champs, tandis qu'elles se valent à peu près pour 3 ou 4, selon la trace exacte considérée.

Le graphique de la Figure 4.35 présentant le score F pour les découpages courts nous montre clairement que les performances décroissent légèrement lorsque le RDM augmente, mais que le NVC a une influence quasi négligeable sur les performances de BaNet3F.

4.3.4 Influence du nombre maximal de champs par trame sur les performances de BaNet3F

On peut voir sur le graphique de la Figure 4.36, représentant le score F lissé pour tous les découpages, que, comme pour l'essentiel des courbes précédentes, les performances restent globalement constantes (légère décroissance pour les valeurs de NMCT importantes) quel que soit le RDM. On constate également que les performances croissent avec le NMCT jusqu'à

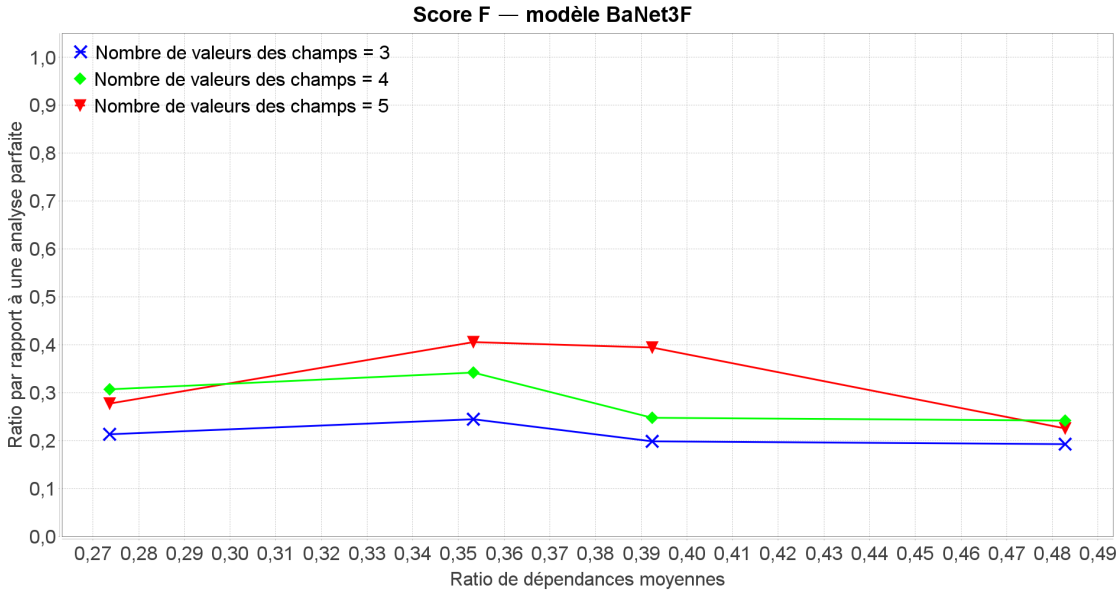


FIGURE 4.33 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NVC, moyenné sur tous les découpages, et lissé sur 3 points

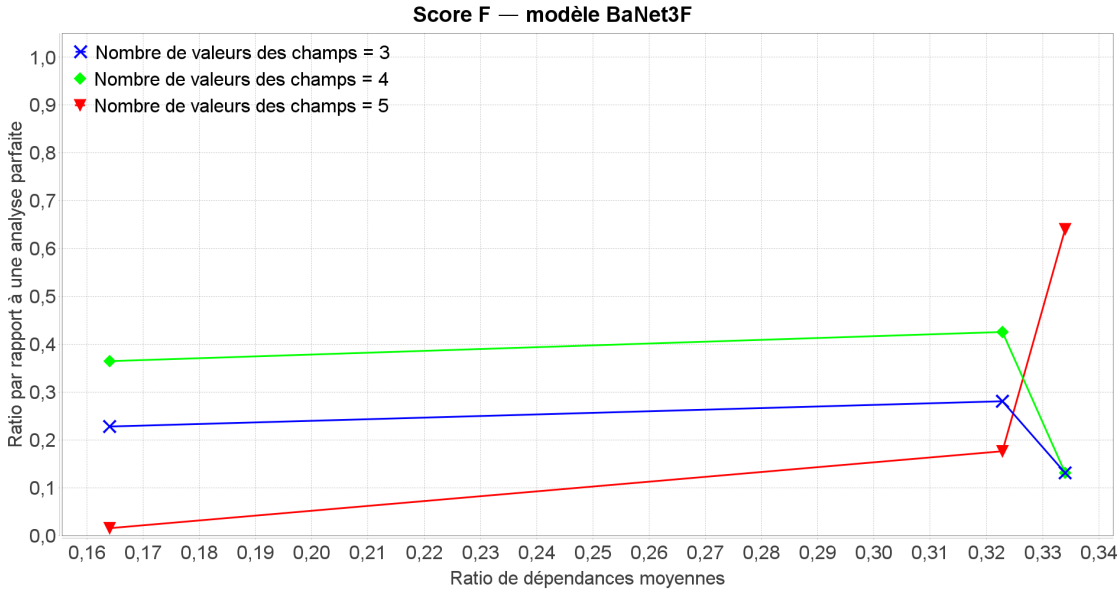


FIGURE 4.34 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NVC, moyenné sur les découpages longs

une valeur de 4, puis décroissent ensuite.

Le graphique de la Figure 4.37 présente le score F pour les découpages longs, et on peut observer que le comportement des courbes par rapport au RDM varie selon la valeur du NMCT considérée. Pour un NMCT de 2, les performances décroissent de façon monotone. Pour un NMCT de 3, les performances restent globalement constantes, en dehors d'un trou

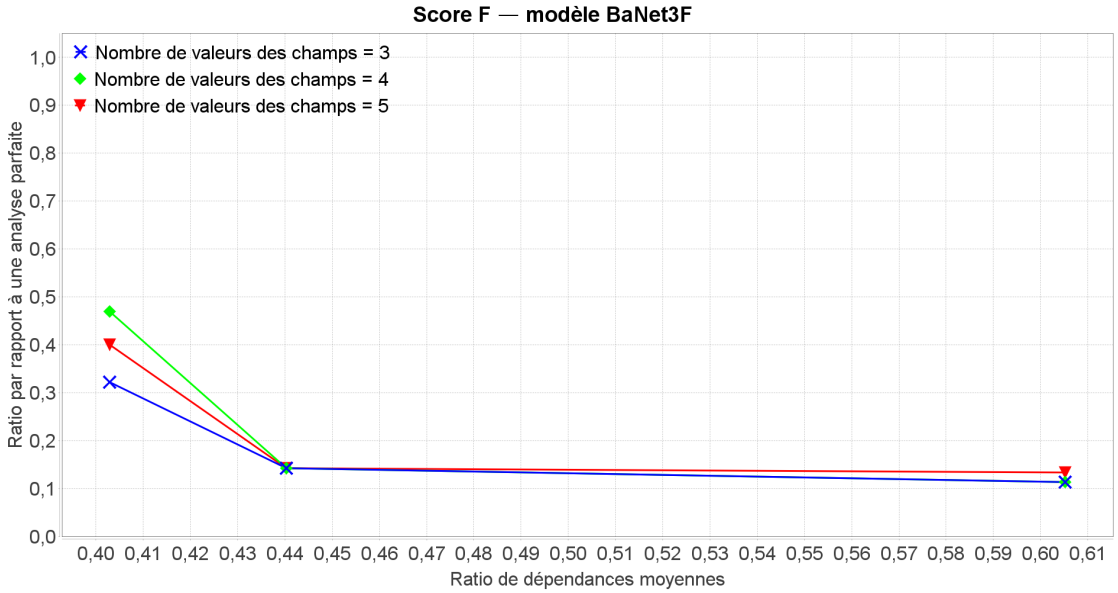


FIGURE 4.35 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NVC, moyenné sur les découpages courts

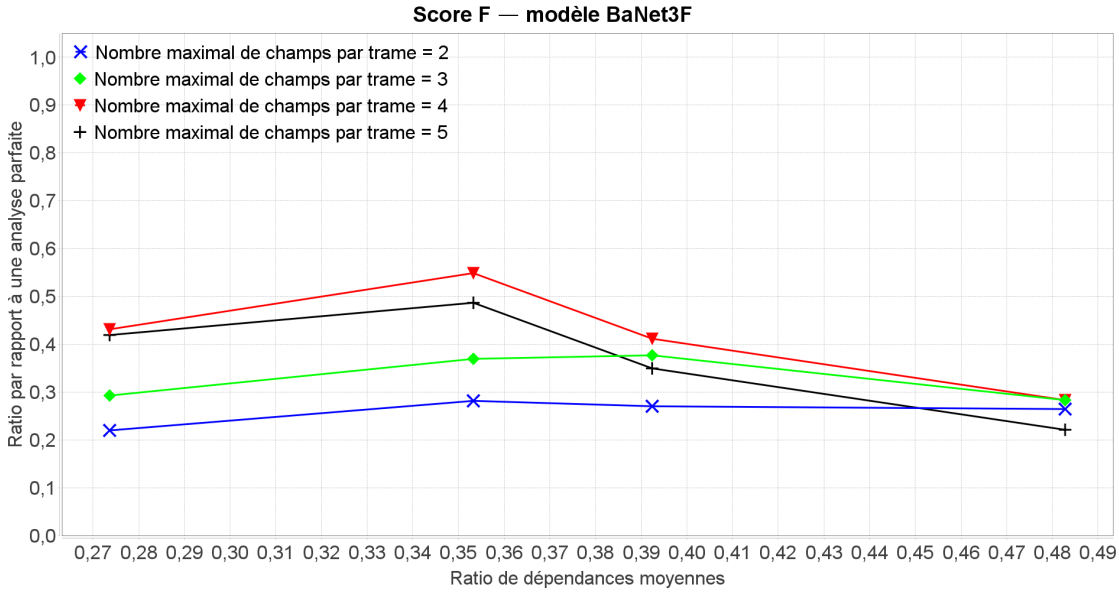


FIGURE 4.36 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur tous les découpages, et lissé sur 3 points

de performances pour l'un des découpages. Enfin, pour un NMCT de 4 ou 5, les performances augmentent globalement. Du point de vue du NMCT, les performances augmentent dans l'ensemble lorsque ce dernier croit.

On constate sur le graphique de la Figure 4.38, présentant le score F pour les découpages courts, que, tout comme pour le NVC, le NMCT a une influence quasi négligeable sur les

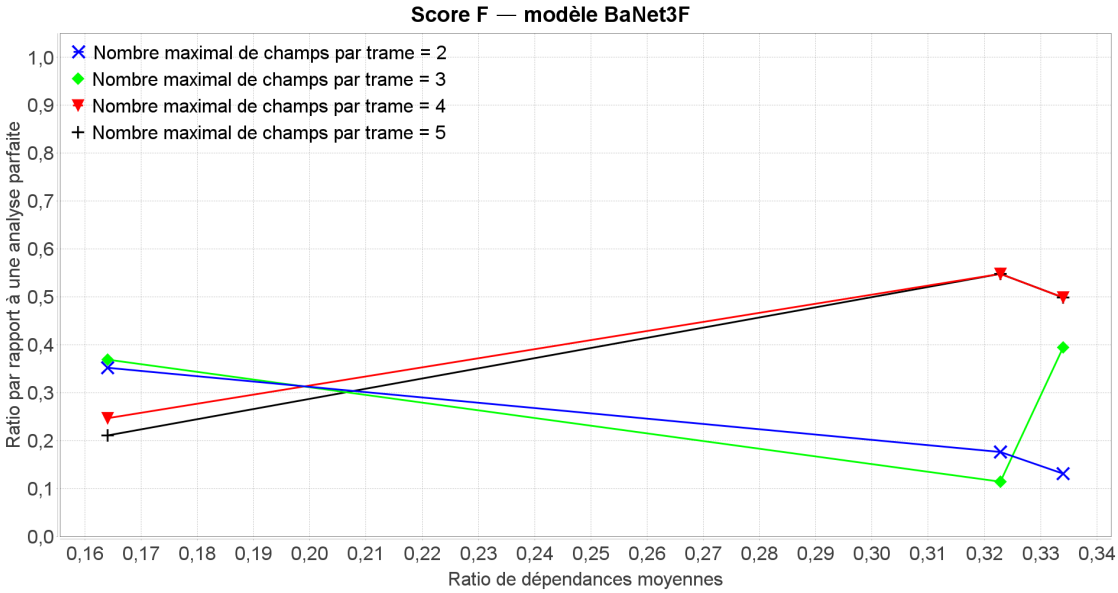


FIGURE 4.37 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages longs

performances de BaNet3F.

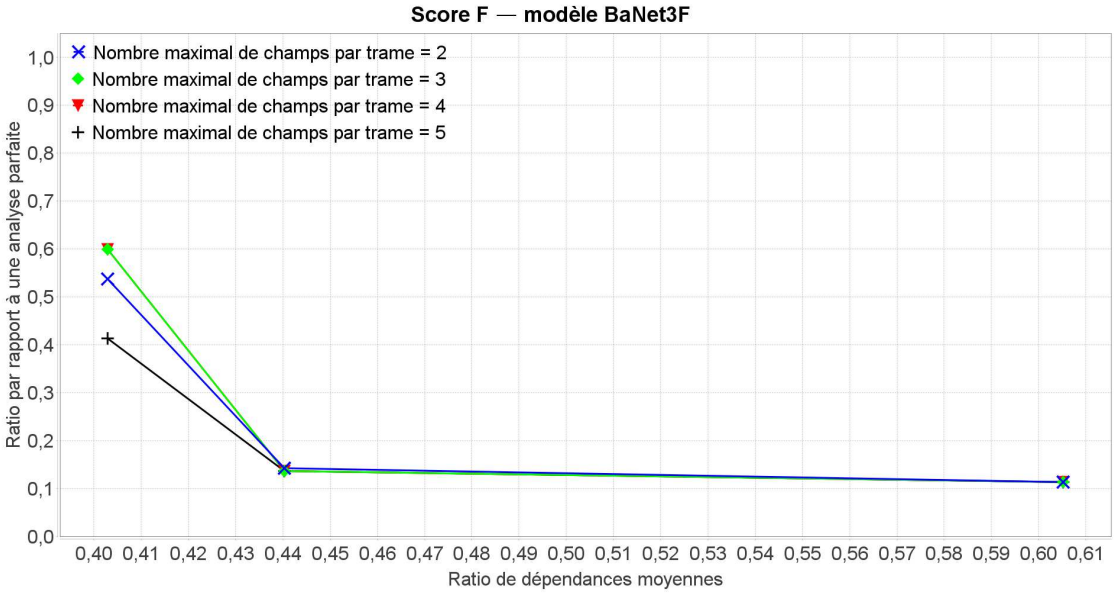


FIGURE 4.38 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMCT, moyenné sur les découpages courts

4.3.5 Influence du seuil relatif de validation du découpage sur les performances de BaNet3F

Le graphique de la Figure 4.39 présente la précision lissée pour tous les découpages, et on peut observer que les performances croissent avec le RDM. On constate également que les performances croissent avec le SRVD à partir de 0.2 et jusqu'à une valeur de 0.4, et qu'elles décroissent ensuite jusqu'à 0.6. Ce comportement peut s'expliquer par le fait que pour les valeurs faibles du SRVD, le découpage récursif des trames par le modèle s'arrête trop tôt, avant de trouver beaucoup de champs, et lorsque le seuil devient trop élevé, au contraire, le découpage récursif se poursuit trop loin, causant la création d'une multitude de petits champs, dont seul quelques-uns sont corrects, d'où la baisse de la précision.

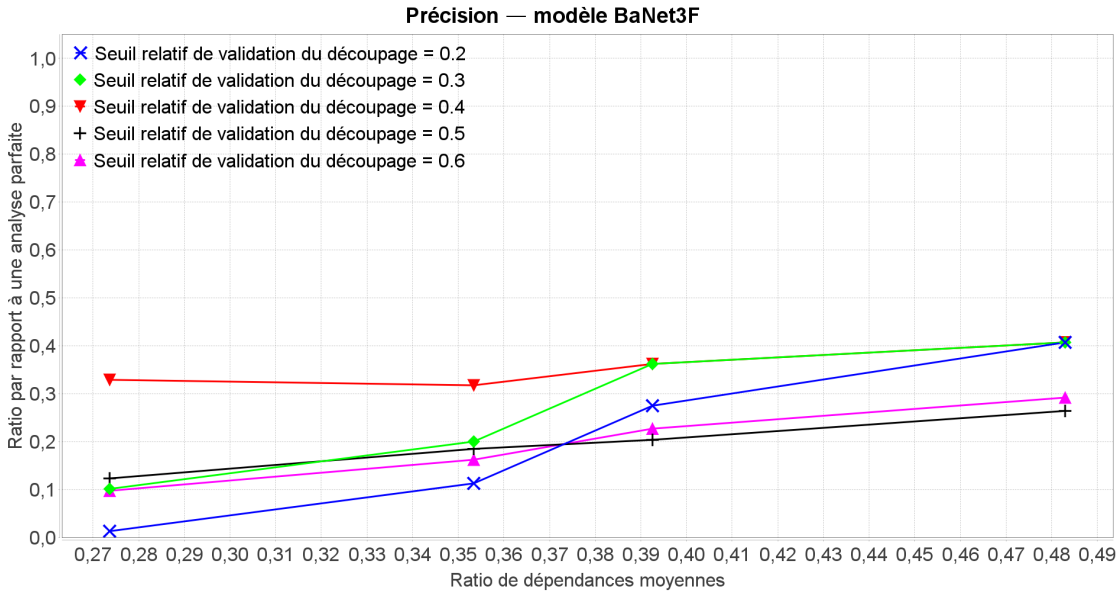


FIGURE 4.39 – Précision du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur tous les découpages, et lissé sur 3 points

On constate sur le graphique de la Figure 4.40, présentant la couverture lissée pour tous les découpages, que les performances croissent avec le RDM comme sur le graphique précédent. Cependant, concernant le comportement relativement au SRVD, on remarque que les performances croissent lorsque ce dernier augmente. Ceci s'explique par le fait que la plupart des champs sont de petite taille, si bien que même si le découpage est trop important dans l'ensemble, créant ainsi une multitude de petits champs, ces derniers contiennent une bonne partie des champs à trouver, ce qui améliore la couverture.

Le graphique de la Figure 4.41 présentant le score F lissé pour tous les découpages, nous montre qu'au final, un SRVD de 0.6 est le meilleur compromis entre précision et couverture.

Sur le graphique de la Figure 4.42, présentant la précision pour les découpages longs, on peut voir que les performances sont globalement constantes quel que soit le RDM. Concernant l'influence du SRVD, on constate un comportement identique à celui du graphique de la

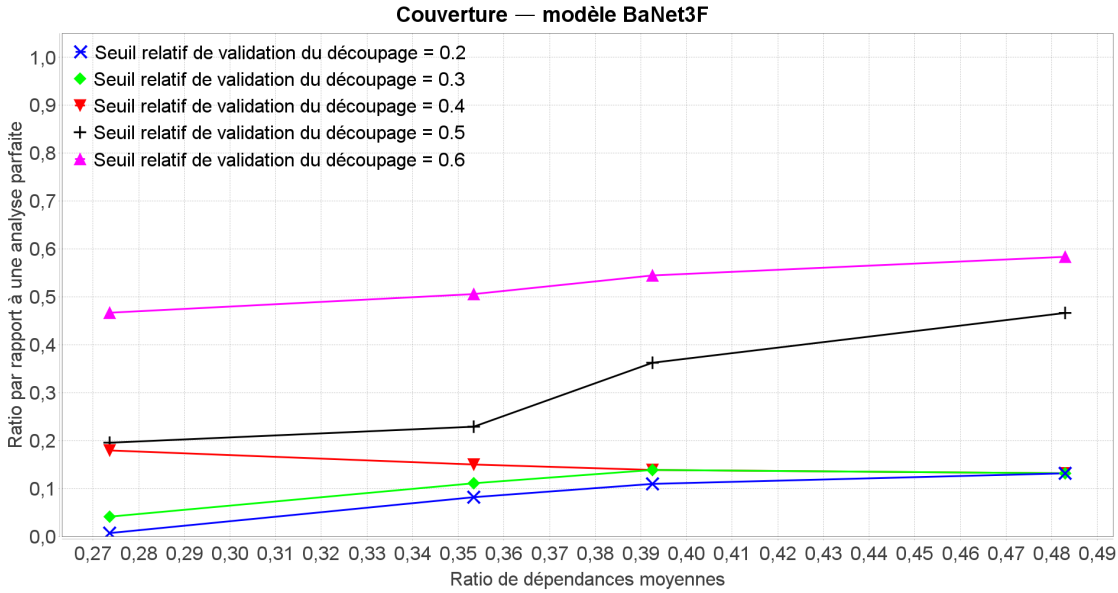


FIGURE 4.40 – Couverture du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur tous les découpages, et lissé sur 3 points

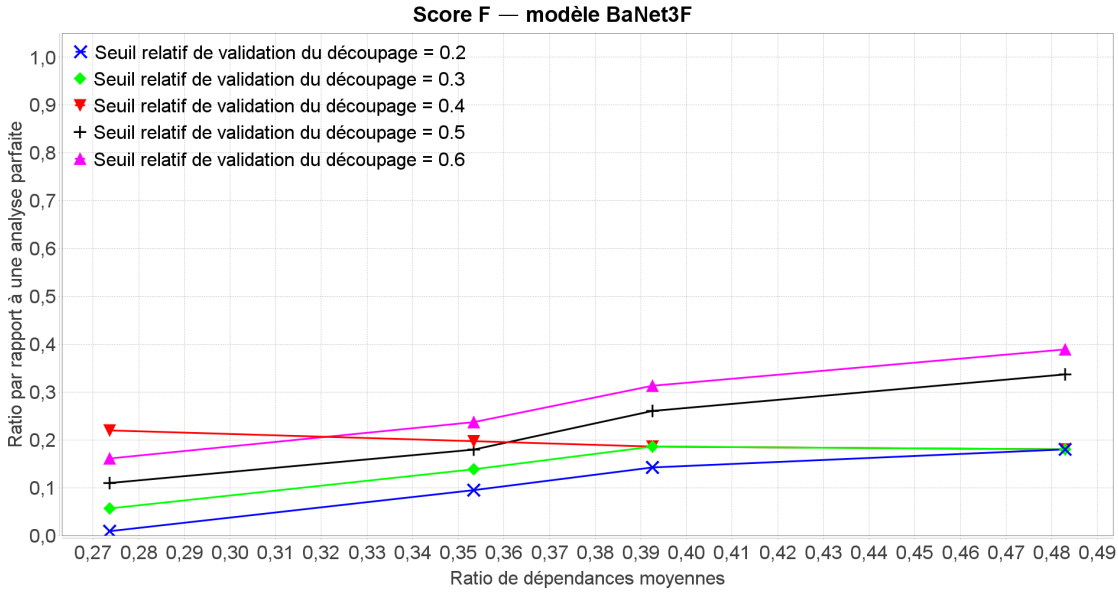


FIGURE 4.41 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur tous les découpages, et lissé sur 3 points

Figure 4.39, à savoir que 0.4 donne les meilleures performances, et qu'au-delà et en-deçà, elles décroissent.

On observe, sur le graphique de la Figure 4.43 présentant la couverture pour les découpages longs, que pour un SRVD de 0.4 et 0.5, les performances décroissent avec l'augmentation du RDM, tandis qu'elles restent constantes pour les autres valeurs. On remarque, de plus,

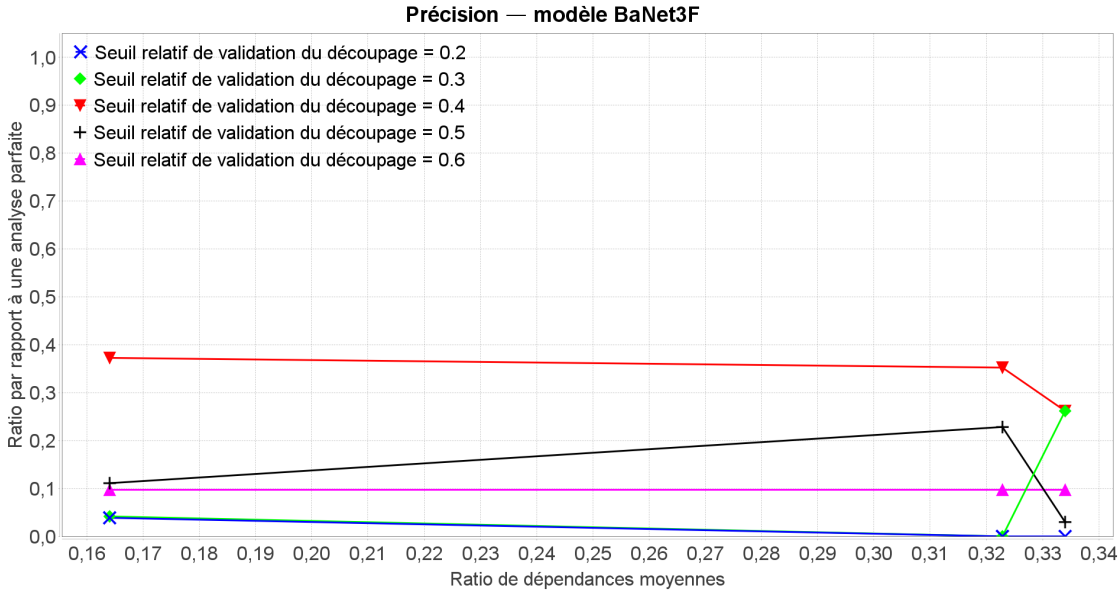


FIGURE 4.42 – Précision du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages longs

que les performances croissent avec l'augmentation du SRVD, comme sur le graphique de la Figure 4.40.

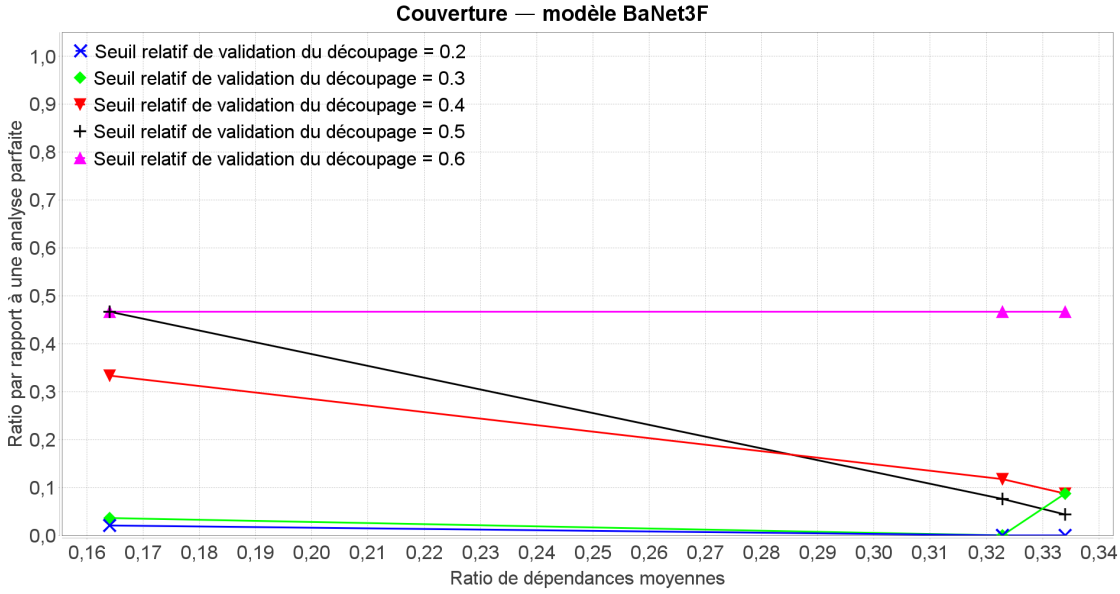


FIGURE 4.43 – Couverture du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages longs

Le graphique de la Figure 4.44, nous montre que le meilleur compromis (score F) pour les découpages longs est une valeur de SRVD de 0.4.

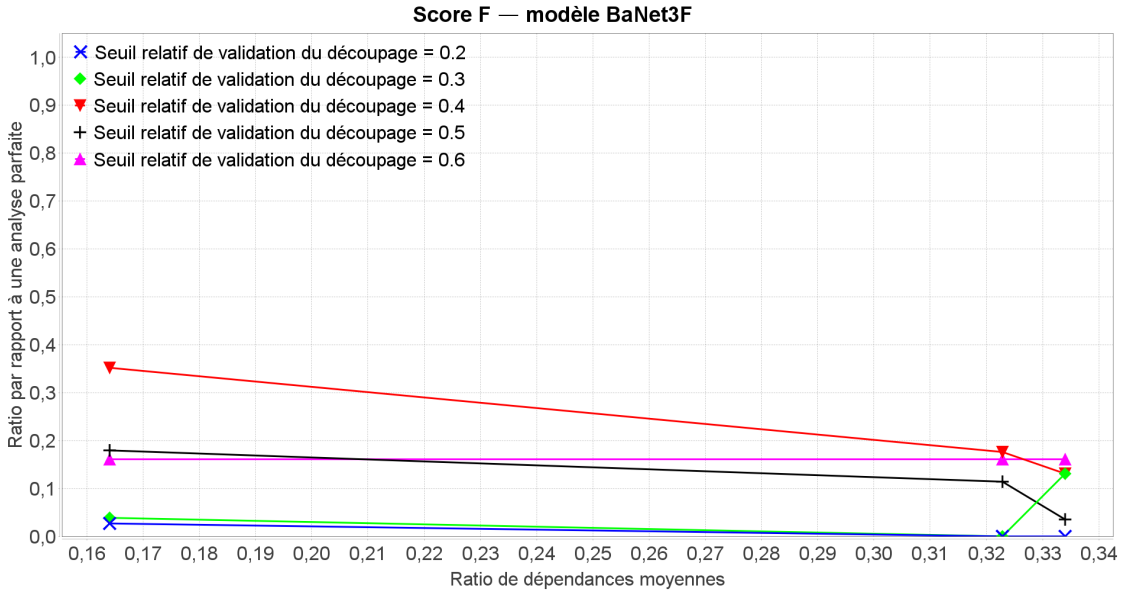


FIGURE 4.44 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages longs

On constate sur le graphique de la Figure 4.45, présentant la précision pour les découpages courts, que les performances restent à peu près constantes quel que soit le RDM. On peut aussi voir que les performance baissent d'un cran lorsque le SRVD dépasse les 0.4. L'absence de diminution des performances lorsque le seuil diminue est probablement due au fait que, contrairement au cas des découpages longs, dès le premier découpage, la probabilité de trouver un champ correct est importante.

Le graphique de la Figure 4.46 présente le score F pour les découpages courts, et on peut voir que les performances restent constantes ou décroissent légèrement lorsque le RDM augmente. On observe que les performances croissent avec l'augmentation du SRVD, quand celui-ci dépasse les 0.4. Cela s'explique par le fait que les découpages courts sont constitués quasi-exclusivement de champs très courts (1 ou 2 bits), si bien que les performances n'entament pas leur croissance tant que le seuil n'est pas suffisamment élevé pour que le découpage récursif se poursuive jusqu'à ce niveau de granularité.

On peut voir sur le graphique de la Figure 4.47, que, comme pour le graphique de la Figure 4.41, la valeur du SRVD constituant le meilleur compromis est de 0.6.

4.3.6 Influence du nombre maximal d'unités élémentaires considérées simultanément sur les performances de BaNet3F

Le graphique de la Figure 4.48 présente le score F lissé pour tous les découpages, et on peut y voir que pour un NMUECS de 2 ou 4, les performances sont globalement constantes quel que soit le RDM (lorsqu'elles sont définies), tandis que pour 8 unités élémentaires

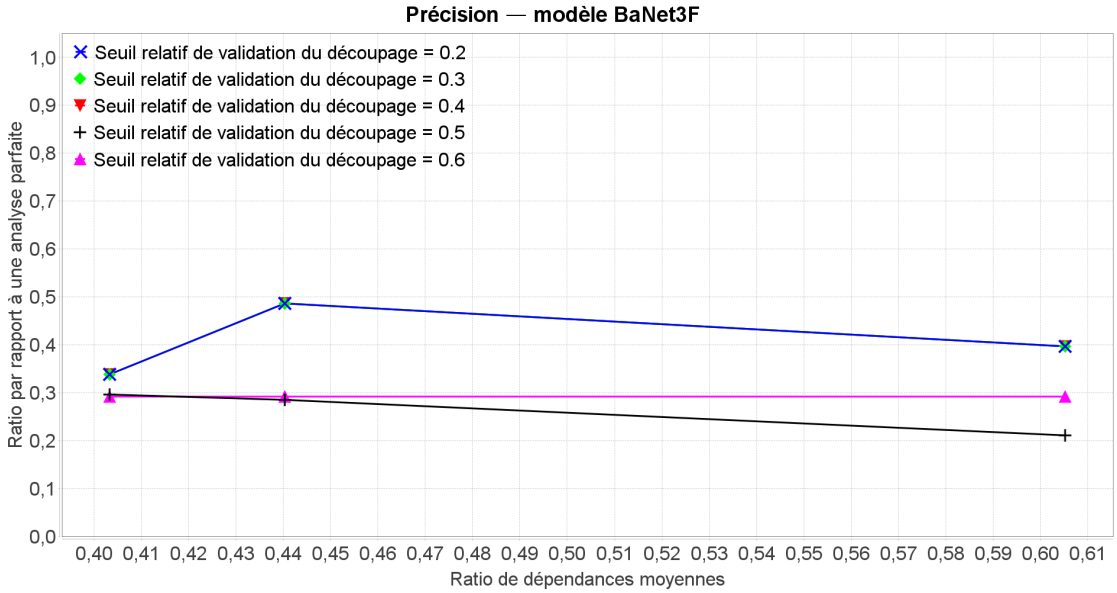


FIGURE 4.45 – Précision du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages courts

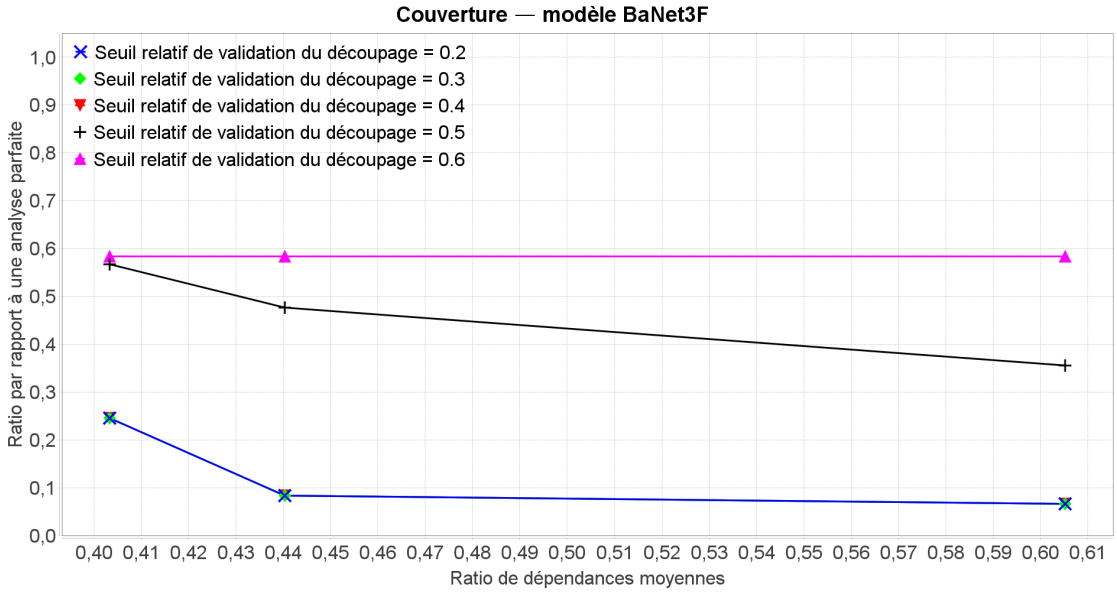


FIGURE 4.46 – Couverture du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages courts

considérées simultanément, les performances décroissent. On peut également remarquer que les performances pour 2 et 4 unités élémentaires considérées simultanément sont à peu près équivalentes, tandis que pour 8, elles sont nettement supérieures.

On peut voir sur le graphique de la Figure 4.49 présentant le score F pour les découpages longs, que les performances croissent légèrement lorsque le NMUECS est de 2 ou 8, tandis

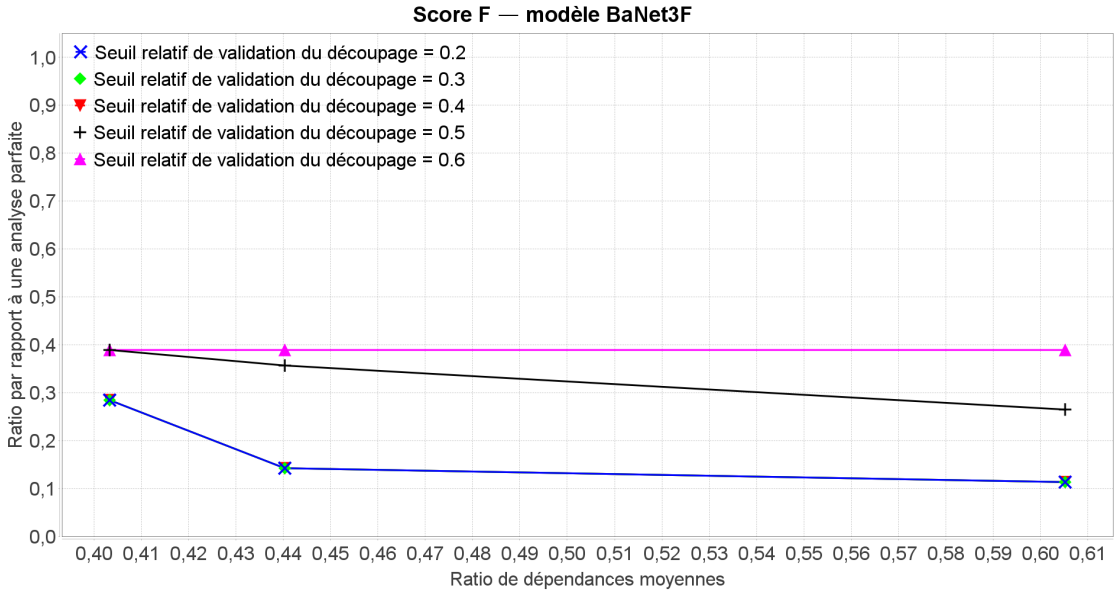


FIGURE 4.47 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le SRVD, moyenné sur les découpages courts

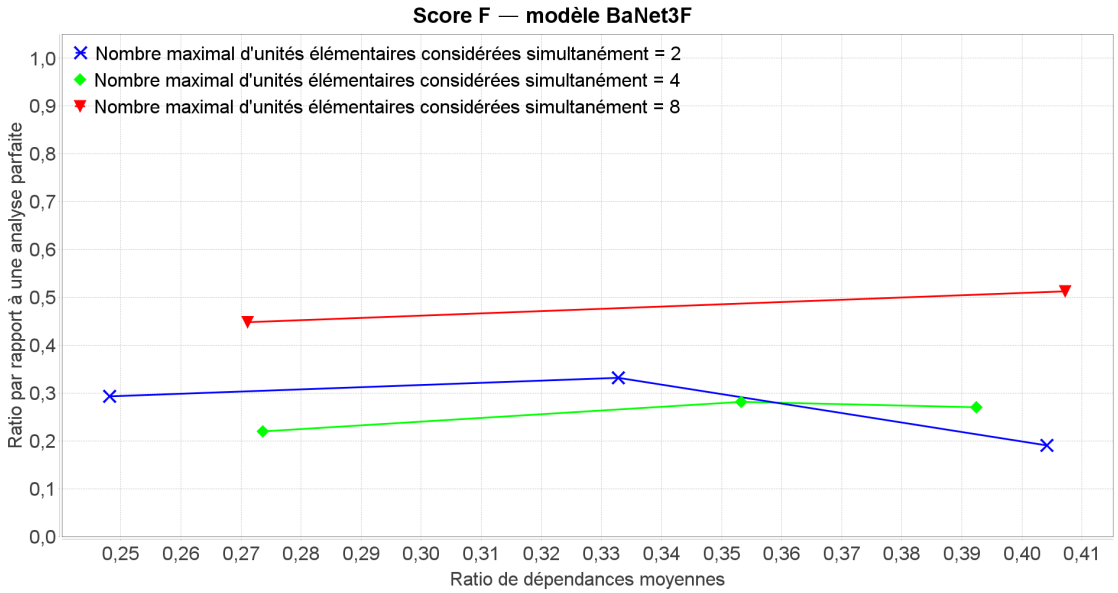


FIGURE 4.48 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur tous les découpages, et lissé sur 3 points

qu'elles décroissent lorsqu'il est de 4. Ce comportement qui peut sembler étrange ne peut être expliqué que par les spécificités des traces analysées, indépendamment des paramètres maîtrisés, comme c'est le cas pour l'essentiel des courbes de cette partie. Comme pour les courbes du graphique de la Figure 4.48, les performances pour 2 et 4 unités élémentaires considérées simultanément ont des performances globalement équivalentes, tandis qu'elles sont nettement plus élevées pour 8.

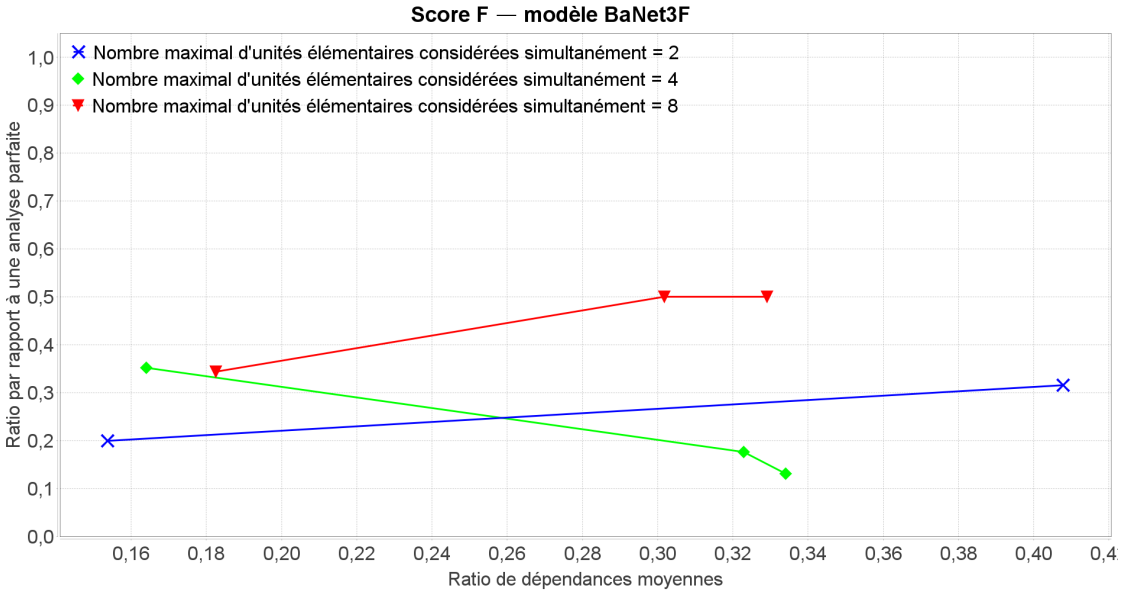


FIGURE 4.49 – Score F du modèle BaNet3F en fonction du RDM et paramétré par le NMUECS, moyenné sur les découpages longs

Le graphique de performances de BaNet3F appliqué aux découpages courts n'étant pas suffisamment exploitable, nous ne le présenterons pas.

4.3.7 Bilan

Nous avons pu constater que les performances de l'analyse des traces réelles par BaNet3F n'affichent aucune tendance claire relativement au ratio de dépendances moyennes, croissant ou décroissant légèrement, ou bien restant constantes. Nous en concluons donc que sur ces traces réelles, cette métrique n'est pas un indicateur suffisant pour donner une idée du niveau de performances attendu, et que cela diverge donc de nos attentes a priori.

Le nombre maximal de valeurs différentes de l'observation ne semble avoir que peu d'influence sur les performances, entraînant juste une baisse lorsqu'il est trop faible (inférieur à 5). Cela ne correspond que partiellement à nos attentes ; nous nous attendions en effet à ce que les performances augmentent avec le NMVDO, mais ceci n'est valable qu'en-dessous de 5.

Les performances de BaNet3F semblent globalement croitre avec l'augmentation du nombre de valeurs des champs, avec une saturation à partir de 5. Ce comportement coïncide bien avec nos attentes, à l'exception de la présence de saturation.

On constate la présence d'une valeur optimale du nombre maximal de champs par trame de 4, ce qui rentre dans le cadre de nos a priori sur l'influence de ce paramètre sur les performances.

Le comportement des performances relativement au seuil relatif de validation du découpage correspond globalement à ce à quoi nous nous attendions, à savoir la présence d'une valeur optimale. Cependant, le fait que les performances restent maximale au-delà de 0.6 est un peu inattendu, sans pour autant être inexplicable (cf commentaires des courbes).

Les performances de BaNet3F semblent augmenter globalement avec le nombre maximal d'unités élémentaires considérées simultanément, même s'il est difficilement possible de l'affirmer avec seulement 3 valeurs différentes, et des courbes pour 2 et 4 très proches. On peut néanmoins dire que cela reste cohérent avec notre hypothèse initiale sur l'influence de ce paramètre, à savoir qu'une valeur plus importante de nombre maximal d'unités élémentaires considérées simultanément améliore les performances.

On ne peut pas vraiment voir de différence dans le comportement des performances de BaNet3F entre les découpages courts et longs.

Enfin, on peut remarquer que globalement, les performances du modèle sur les traces réelles sont plus faibles que celles sur les traces générées, alors même que les valeurs du ratio de dépendances moyennes des deux types de traces sont comparables. En effet, avec les traces générées, les performances varient entre parfaites (100%) et très faibles ($\sim$ 10%), tandis qu'avec les traces réelles, elles varient entre moyennes ($\sim$50%) et très faibles ($\sim$10%). Cependant, on peut observer que les traces réelles possèdent globalement des valeurs de ratio de dépendances moyennes plus élevées que celles des traces générées. Si l'on considère donc les performances à ratio de dépendances moyennes équivalent, on constate que celles-ci sont environ équivalentes sur trames générées et réelles (30-50% dans les deux cas), comme illustré sur le graphique de la Figure 4.50.

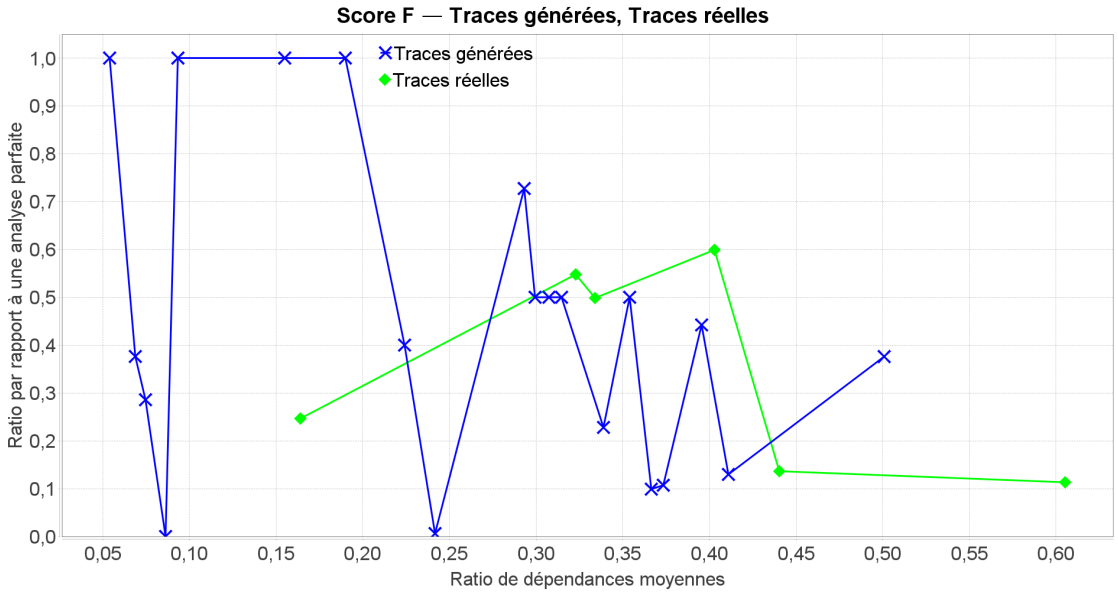


FIGURE 4.50 – Comparaison du Score F du modèle BaNet3F en fonction du RDM entre traces réelles et générées, moyenné sur tous les découpages

Cette conjecture ne semble toutefois pas s'appliquer aux traces réelles dont la valeur du RDM est relativement faible (< 0.25), pour lesquelles les performances sont beaucoup plus faibles que sur traces générées. Nous nous emploierons à trouver des pistes permettant d'expliquer cette différence de performance dans la suite de ce chapitre.

4.4 Explication de la différence de performances entre traces réelles et générées

En observant les traces générées et les traces réelles, nous avons constaté un certain nombre de différences importantes. Nous avons donc supposé que celles-ci étaient susceptibles d'expliquer la baisse des performances de BaNet3F dans certains cas lorsque appliqué aux traces réelles. Afin de s'assurer de cela, nous avons formulé six hypothèses sur des différences indépendantes les unes des autres, et cherché à les tester chacune. Pour chaque hypothèse, nous avons généré une trace semblable en tout point aux traces générées de référence présentées au début de ce chapitre (nous avons choisi le découpage [3,1,1,1,1,1,1,1,2,2,2,8,16,16,16]), à une légère modification près sur le plan de cette hypothèse, de façon à se rapprocher d'une trace réelle. Ensuite, nous avons évalué les performances de BaNet3F sur ces traces, et les avons comparées aux meilleures performances obtenues sur les traces générées de référence. Celles-ci sont déterminées en cherchant celles avec le plus proche ratio de dépendances moyennes, et si plusieurs sont proches, alors celles avec le découpage le plus proche (en l'occurrence, celles incluant des découpages mêlant bits et octets) est choisie. À partir de cela, nous concluons sur la confirmation ou l'infirmation de chaque hypothèse, et tentons de trouver une explication logique à ce comportement.

4.4.1 Hypothèse 1 : différence dans la diversité des valeurs possibles des champs

Le nombre de valeurs possibles de chaque champ des trames des traces générées est assez faible, variant de manière effective entre 2 et 6, dans l'essentiel des cas. Cependant, la diversité des valeurs possibles des champs des traces réelles est nettement plus importante, s'élevant à plus d'une dizaine de valeurs différentes, notamment pour les champs correspondant à des adresses. Nous émettons donc l'hypothèse qu'une diversité importante réduit les performances de BaNet3F.

Afin de tester cette hypothèse, nous avons augmenté le **nombre maximum de valeurs d'un champ** de 10 à 20, et supprimé la limitation des valeurs possibles à celles possédant une probabilité suffisamment importante. Ainsi, dans la Trace 1, au lieu de 6 valeurs maximum possibles, il y en a 20.

Cette première trace possède un RDM de 0.34 environ, et les meilleures performances (score F) correspondantes obtenues sur traces générées sont d'environ 37%, qui nous servira donc de référence. Les meilleures performances que nous avons réussi à obtenir avec cette trace sont de 32%, ce qui, aux aléas de génération près, correspond donc à des performances équivalentes à celles obtenues sur traces générées. Nous concluons donc que cette hypothèse est infirmée, à savoir que le nombre des valeurs des champs ne semble pas avoir d'impact significatif sur les performances de BaNet3F.

4.4.2 Hypothèse 2 : différence dans la répartition statistique des valeurs possibles des champs

Les lois de génération des valeurs des champs des trames générées sont conçues de façon à ce que certaines valeurs soient plus présentes que d'autres, mais en restant dans un rapport entre les plus fortes probabilités et les plus faibles ne dépassant pas 10. Toutefois, pour les trames générées, ce rapport peut s'élever à 100, voir 1000. Nous formulons donc l'hypothèse qu'une forte prédominance de certaines valeurs par rapport aux autres fait baisser les performances de BaNet3F.

Afin de tester cette hypothèse, nous avons augmenté le **facteur de la distribution intra-champs** de 5 à 10, et réduit la limitation des valeurs possibles à celles possédant une probabilité suffisamment importante. Ainsi, dans la Trace 2, le rapport entre les plus fortes probabilités et les plus faibles s'élève à environ 200.

Cette seconde trace possède un RDM de 0.37 environ, et les meilleures performances (score F) correspondantes obtenues sur traces générées sont d'environ 29%, qui nous servira donc de référence. Les meilleures performances que nous avons réussi à obtenir avec cette trace sont de 35%, ce qui, aux aléas de génération près, correspond donc à des performances équivalentes à celles obtenues sur traces générées. Nous concluons donc que cette hypothèse est infirmée, à savoir que la prédominance de quelques valeurs par rapport aux autres au sein d'un champ ne semble pas avoir d'impact significatif sur les performances de BaNet3F.

4.4.3 Hypothèse 3 : différence dans le nombre de bits fixes

Nous avons conçu le générateur de trames de façon à ce qu'aucun bit d'aucun champ n'ait jamais tout le temps la même valeur dans les traces générées. Or, dans les traces réelles, beaucoup de bits (comprendre position de bit) ont tout le temps la même valeur, quelle que soit la valeur des champs auxquels ils appartiennent. Nous faisons donc l'hypothèse que la présence de bits invariants entraîne une diminution des performances de BaNet3F.

Afin de tester cette hypothèse, nous avons introduit un décalage des distributions intra-champs, de façon à ce qu'aucune valeur des champs ne soit dans la deuxième moitié des valeurs théoriques possibles (combinaisons possibles) étant donné la longueur des champs. Ainsi, dans la Trace, pour chaque champ, seuls les 2 derniers bits prennent des valeurs différentes selon les trames, tous les autres restant strictement identiques (quand ils existent).

Cette troisième trace possède un RDM de 4.07 environ, et les meilleures performances (score F) correspondantes obtenues sur traces générées sont d'environ 24%, qui nous servira donc de référence. Cependant, cette valeur est obtenue sur une trace dont le RDM est d'environ 0.41, ce qui constitue un écart suffisamment important pour rendre la comparaison entre les valeurs de performances sur cette trace et les traces générées difficilement légitime. Les meilleures performances que nous avons réussi à obtenir avec cette trace sont de 20%, ce qui, aux aléas de génération près, signifie que les performances sur cette trace sont équivalentes à celles obtenues sur traces générées. Nous concluons donc que cette hypothèse est infirmée, à savoir que la présence de bits invariants au sein des champs ne semble pas avoir d'impact significatif sur les performances de BaNet3F.

4.4.4 Hypothèse 4 : différence dans le niveau de dépendance entre champs

Dans les traces générées, quelle que soit la valeur d'un champ, les valeurs possibles des champs suivants sont les mêmes, seules leurs probabilités changent. Cela signifie que la dépendance entre deux valeurs de champ est toujours finie. Cependant, dans les traces réelles, l'apparition de certaines valeurs de champs entraîne la certitude de l'apparition ou de l'absence de valeurs de champs subséquents. C'est à dire qu'il existe une dépendance infinie entre des valeurs de certains champs. Nous émettons donc l'hypothèse qu'une dépendance infinie réduit les performances de BaNet3F.

Afin de tester cette hypothèse, nous avons inséré un couplage total de certaines valeurs de champs. Ainsi, dans la Trace 4, pour certains champs, la présence d'une valeur entraîne systématiquement l'apparition d'une autre dans certains champs subséquents.

Cette quatrième trace possède un RDM de 4.64 environ, et les meilleures performances (score F) correspondantes obtenues sur traces générées sont d'environ 24%, qui nous servira donc de référence. Cependant, on rencontre le même problème que pour l'hypothèse précédente, notre conclusion est donc à considérer avec réserve. Les meilleures performances que nous avons réussi à obtenir avec cette trace sont de 30%, ce qui, aux aléas de génération près, signifie que les performances sur cette trace sont équivalentes à celles obtenues sur traces générées. Nous concluons donc que cette hypothèse est infirmée, à savoir que le couplage total de certaines valeurs de champs ne semble pas avoir d'impact significatif sur les performances de BaNet3F.

4.4.5 Hypothèse 5 : différence dans le nombre de champs de 1 bit indépendants

De par la conception même du générateur, les champs de 1 bit se retrouvent à être tous indépendants les uns des autres. Toutefois, dans les traces réelles, il existe bel et bien une dépendance des champs de 1 bits avec les précédents. Nous formulons donc l'hypothèse que la présence de dépendance dans les champs de 1 bit fait baisser les performances de BaNet3F.

Afin de tester cette hypothèse, nous avons introduit des dépendances aléatoires entre chaque champ de 1 bit et le champ le précédant, tout en conservant la possibilité pour chaque valeur d'apparaître. Ainsi, dans la Trace 5, chaque champ de 1 bit est dépendant du champ le précédant.

Cette seconde trace possède un RDM de 0.35 environ, et les meilleures performances (score F) correspondantes obtenues sur traces générées sont d'environ 37%, qui nous servira donc de référence. Les meilleures performances que nous avons réussi à obtenir avec cette trace sont de 31%, ce qui, aux aléas de génération près, correspond donc à des performances équivalentes à celles obtenues sur traces générées. Nous concluons donc que cette hypothèse est infirmée, à savoir que la dépendance des champs de 1 bits consécutifs les uns aux autres ne semble pas avoir d'impact significatif sur les performances de BaNet3F.

4.4.6 Hypothèse 6 : différence dans le nombre de valeurs possibles des champs de 1 bit

Les champs de 1 bit des trames générées peuvent toujours prendre les 2 valeurs possibles (0 et 1). Or, dans les trames réelles, certains champs ont systématiquement la même valeur, comme les bits non utilisés des champs réservés pour un potentiel usage ultérieur, ou les flags caractéristiques du type de communication choisi au sein du réseau observé. Nous faisons donc l'hypothèse que la présence de champs ayant toujours la même valeur au sein d'une trace entraîne une diminution des performances de BaNet3F.

Afin de tester cette hypothèse, nous avons fixé la probabilité d'apparition de certaines valeurs de champs de 1 bit à 1. Ainsi, dans la Trace 6, ces champs prendront systématiquement la même valeur dans toute la trace.

Cette sixième trace possède un RDM de 0.2 environ, et les meilleures performances (score F) correspondantes obtenues sur traces générées sont d'environ 90%, qui nous servira donc de référence. Les meilleures performances que nous avons réussi à obtenir avec cette trace sont de 56%, ce qui, même en considérant les aléas de génération, signifie donc que les performances obtenues sur cette trace sont inférieures à celles obtenues sur traces générées. Nous concluons donc que cette hypothèse est confirmée, à savoir que la présence de champs à la valeur invariantes entraîne une diminution des performances de BaNet3F.

Ce résultat n'est guère surprenant, car si des champs prennent constamment la même valeur, alors qu'ils soient considérés séparément ou rattachés aux champs voisins, la probabilité globale du champ dont la valeur est fixe et ses voisins reste la même. Le modèle n'a donc aucun moyen de savoir s'il doit les séparer ou non, la décision prise est donc aléatoire, ce qui a pour effet de faire baisser les performances.

4.4.7 Bilan sur les hypothèses testées

La Table 4.5 récapitule les hypothèses formulées, les données numériques associées, et les conclusions quant à leur confirmation ou infirmation.

Finalement, 5 des 6 hypothèses formulées ont été infirmées ; la seule raison que nous ayons trouvé qui semble expliquer la baisse des performances entre traces générées et réelles dans certains cas est la présence de champs aux valeurs fixes parmi ceux de petite taille des deux premiers octets des trames.

On peut également remarquer que la seule différence de performances notable observée entre les traces de test des hypothèses et les traces générées se manifeste lorsque les performances sur traces générées sont fortes et/ou que le ratio de dépendances moyennes est relativement faible, ce qui coïncide bien avec nos observations initiales.

Hypothèse	Ratio de dépendances moyennes	Score F de référence	Score F obtenu	Décision
1 : Différence dans la diversité des valeurs possibles des champs	0.34	37%	32%	Infirmé
2 : Différence dans la répartition statistique des valeurs possibles des champs	0.37	29%	35%	Infirmé
3 : Différence dans le nombre de bits fixes	4.07	24%	20%	Infirmé
4 : Différence dans le niveau de dépendance entre champs	4.64	24%	30%	Infirmé
5 : Différence dans le nombre de champs de 1 bit indépendants	0.35	37%	31%	Infirmé
6 : Différence dans le nombre de valeurs possibles des champs de 1 bit	0.2	90%	56%	Confirmé

TABLE 4.5 – Synthèse des hypothèses visant à expliquer les différences de performances entre traces générées et réelles

4.5 Conclusion

Nous avons pu voir dans ce chapitre que les performances sur traces générées de BaNet3F sont très bonnes (>80%), voire parfaites (100%) dans des conditions idéales, c'est à dire lorsque la dépendance entre les champs est faible ou nulle, et que le découpage à trouver est particulièrement simple (2 ou 3 champs et pas de mélange entre champs de quelques bits et de plusieurs octets). Lorsque la complexité de la trace à analyser augmente ($\sim$ le ratio de dépendances moyennes augmente), ou que le découpage n'est pas trouvable uniquement par découpage récursif en 2 parties de longueurs 1,2,3,4,8,16,32, ou 64 bits, les performances descendent à un niveau moyen-faible (10%-50%).

Concernant les traces réelles, les performances sont moyennes-faibles (10%-50%) quel que soit le ratio de dépendances, ce qui paraît logique car ces traces ne sont pas triviales comme certaines des traces générées analysées. Ainsi, à l'exception des traces possédant un faible ratio de dépendances moyennes, les performances obtenues sur traces générées et réelles sont globalement équivalentes.

La seule raison qui nous semble pouvoir expliquer la baisse des performances entre traces générées et traces réelles pour les faibles valeurs de ratio de dépendances moyennes est la présence de champs à valeur fixe. Cependant, nous sommes loin d'avoir fait une recherche extensive sur ce point, il existe donc très probablement d'autres raison permettant d'expliquer ce phénomène.

Dans le prochain chapitre, nous comparerons BaNet3F à deux modèles de l'état de l'art, LDA et le modèle de Cai et al.

Chapitre 5

Performances comparées des modèles BaNet3F, LDA et de Cai et al.

Nous avons vu dans le chapitre précédent quel était globalement le niveau de performance que l'on pouvait attendre de BaNet3F. Nous savons donc que lorsque les trames à analyser ne sont pas trop complexes, on peut espérer de bonnes performances (>80%). Pour cette raison, dans ce chapitre, nous n'évaluerons les performances des différents modèles que sur les traces générées nous ayant permis d'obtenir de bonnes performances, à savoir celles ayant pour découpage [1,2,1,3,1,8,16,32], [16,8,8,32], [3,1,1,1,2], [32,32], et [4,4]. En dehors de cela, les paramètres de génération employés seront identiques à ceux présentés dans le chapitre 4.

Dans ce chapitre, nous chercherons à confronter BaNet3F à deux modèles de l'état de l'art : Latent Dirichlet Allocation et le modèle de Cai et al. [60].

Nous avons déjà vu le modèle LDA dans les chapitres 1 et 2, où nous avons observé qu'il s'agissait du plus performant des trois modèles testés, et il nous a semblé intéressant de voir ce que notre modèle, BaNet3F pouvait apporter comme gain de performance. Cependant, le modèle LDA permet d'identifier des séquences de bits caractéristiques dans un ensemble de trames, tandis que notre modèle permet de trouver le découpage caché d'un ensemble de trames. Nous serons donc obligés de convertir les résultats de BaNet3F afin de les rendre comparables à ceux de LDA.

Le modèle de Cai et al. est celui ayant servi de point de départ au nôtre, il nous a donc semblé également intéressant de le comparer à BaNet3F.

Afin de rendre la comparaison la plus juste possible, toutes les mesures d'amélioration des performances par restriction de l'ensemble des possibles étant utilisées dans BaNet3F sont également introduites dans les modèles LDA et Cai et al., lorsque possible. Nous pensons notamment aux restrictions apportées sur les longueurs possibles des champs.

5.1 Performances comparées des modèles BaNet3F et LDA

5.1.1 Cadre de la comparaison

Comme nous l'avons dit, BaNet3F et LDA ne produisent pas le même type de résultats, nous avons donc dû convertir les champs trouvés par BaNet3F en séquences de bits comparables avec celles trouvées par LDA. Pour ce faire, nous avons simplement considéré chaque valeur prise par chaque champ trouvé comme une séquence trouvée.

Afin d'évaluer les performances des deux modèles, nous utiliserons le score F, calculé comme expliqué dans la section 2.1.3.

La stratégie que nous utilisons pour essayer d'avoir une confrontation la plus juste possible entre les deux modèles consiste à étudier indépendamment chacun des deux modèles appliqué à des ensembles de traces de caractéristiques identiques. Pour chacun de ceux-ci, nous évaluerons l'influence de ses paramètres sur les performances obtenues, et parmi toutes les courbes générées, nous choisirons la meilleure. Ensuite, nous comparerons directement ces deux courbes de performances.

À l'aide des simulations déjà effectuées dans les chapitres 2 et 4, nous avons déterminé les paramètres dont l'étude était intéressante pour chacun des deux modèles. Nous avons de plus constaté qu'Alpha, le paramètre de Dirichlet a priori de la distribution des mots-clés par trame, et le gradient maximal de perplexité ne semblaient pas avoir d'impact sur les performances, nous les ignorerons donc dans ce chapitre.

Pour LDA, les paramètres intéressants sont :
— Le nombre de mots-clés
— Le gradient maximal de probabilité

Pour BaNet3F, les paramètres intéressants sont :
— Le seuil relatif de validation du découpage
— Le nombre maximal de champs par trame

5.1.2 Performances du modèle LDA

La Table 5.1 synthétise les paramètres des campagnes de simulations visant à déterminer la meilleure courbe de performances de LDA pour les traces analysées.

Pour les valeurs du nombre de mots-clefs et du gradient maximal de probabilité utilisées, nous sommes partis des valeurs déjà employées dans les simulations du chapitre 2, et avons a posteriori sélectionné les plus intéressantes.

Chaque point des courbes de performances est calculé en moyennant sur tous les découpages cachés différents, et en effectuant un lissage identique à celui employé dans le chapitre 4.

Paramètre	Campagne 1	Campagne 2
Nombre de trames	1000	1000
Facteur de la distribution intra champ inital	3	3
Facteur de la distribution intra champs	5	5
Facteur de la distribution inter champs	0 ;-1 ;-2 ;-3 ;-4 ;-5	0 ;-1 ;-2 ;-3 ;-4 ;-5
Nombre maximum de valeurs d'un champ	10	10
Découpage caché	4,4 3,1,1,1,2 32,32 16,8,8,32 1,2,1,3,1,8,16,32	4,4 3,1,1,1,2 32,32 16,8,8,32 1,2,1,3,1,8,16,32
Nombre de mots-clefs	2 ;3 ;5 ;10	10
Gradient maximal de probabilité	0.1	0.01 ;0.1 ;1
Nombre de répétitions	10	10

TABLE 5.1 – Paramètres des campagnes de simulation du modèle LDA

Campagne 1 : Nombre de mots-clefs

Sur le graphique de la Figure 5.1, présentant le score F lissé, nous pouvons voir que la meilleure courbe présente des performances maximales s'élevant à environ 35%, pour un nombre de mots-clés de 3.

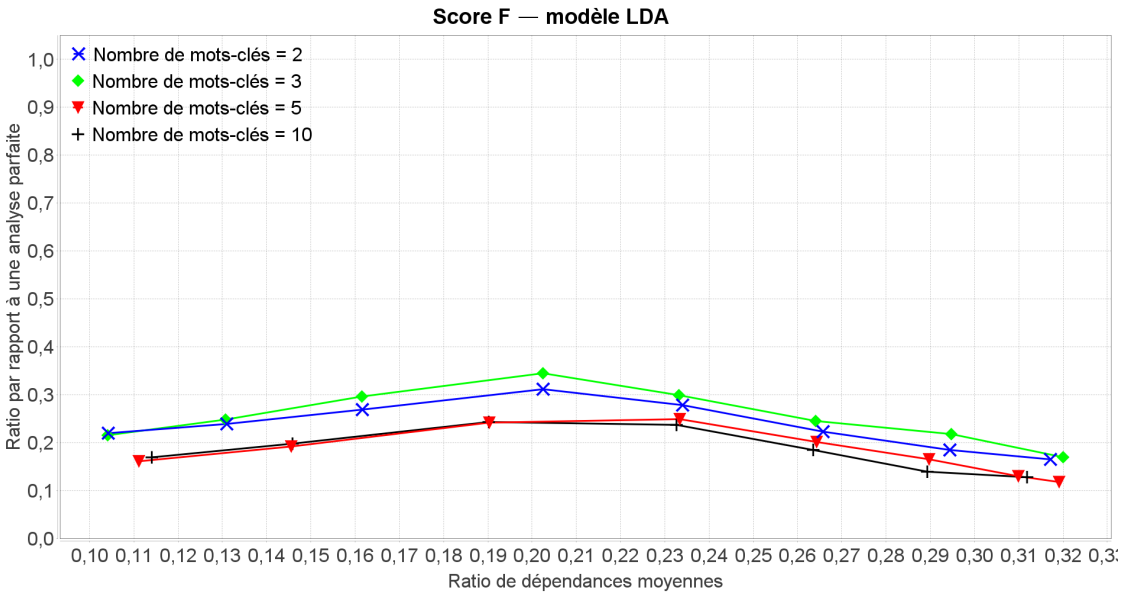


FIGURE 5.1 – Score F du modèle LDA en fonction du ratio de dépendances moyennes, paramétré par le nombre de mots-clefs, et lissé sur 5 points

Campagne 2 : Gradient maximal de probabilité

On observe sur le graphique de la Figure 5.2, présentant le score F lissé, que la meilleure courbe présente des performances maximales s'élevant à environ 25%, pour un gradient maximal de probabilité de 0.1.

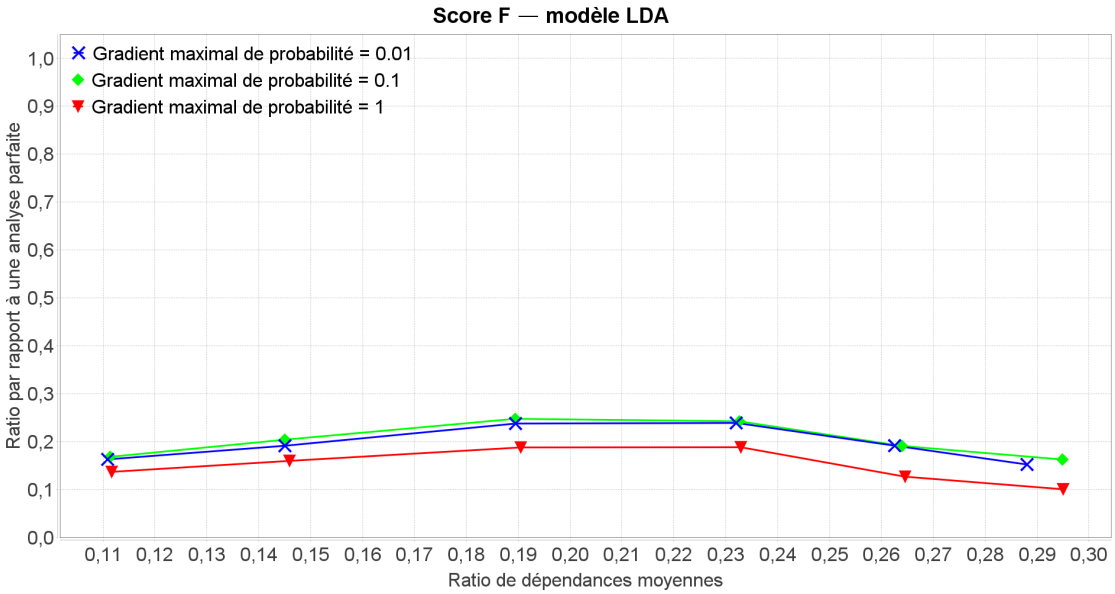


FIGURE 5.2 – Score F du modèle LDA en fonction du ratio de dépendances moyennes, paramétré par le gradient maximal de probabilité, et lissé sur 5 points

La meilleure courbe du modèle LDA est donc celle pour un nombre de mots-clés de 3 et un gradient maximal de probabilité de 0.1.

5.1.3 Performances du modèle BaNet3F

La Table 5.2 synthétise les paramètres des campagnes de simulations visant à déterminer la meilleure courbe de performances de BaNet3F pour les traces analysées.

Pour les valeurs du nombre maximal de champs par trame et du seuil relatif de validation du découpage utilisés, nous avons repris les valeurs déjà employées dans les simulations du chapitre 2.

Chaque point des courbes de performances est calculé en moyennant sur tous les découpages cachés différents, et en effectuant un lissage identique à celui employé dans le chapitre 4.

Campagne 1 : Nombre maximal de champs par trame

On peut constater sur le graphique de la Figure 5.3, présentant le score F lissé, que la meilleure courbe présente des performances s'élevant à environ 72%, pour un nombre maximal

Paramètre	Campagne 1	Campagne 2
Nombre de trames	1000	1000
Facteur de la distribution intra champ inital	3	3
Facteur de la distribution intra champs	5	5
Facteur de la distribution inter champs	0 ;-1 ;-2 ;-3 ;-4 ;-5	0 ;-1 ;-2 ;-3 ;-4 ;-5
Nombre maximum de valeurs d'un champ	10	10
Découpage caché	4,4 3,1,1,1,2 32,32 16,8,8,32 1,2,1,3,1,8,16,32	4,4 3,1,1,1,2 32,32 16,8,8,32 1,2,1,3,1,8,16,32
Nombre maximal de champs par trame	2 ;3 ;4	2
Seuil relatif de validation du découpage	0.2	0.12 ;0.14 ;0.16 ;0.18 ;0.2
Nombre de répétitions	10	10

TABLE 5.2 – Paramètres des campagnes de simulation du modèle BaNet3F

de champs par trame de 4.

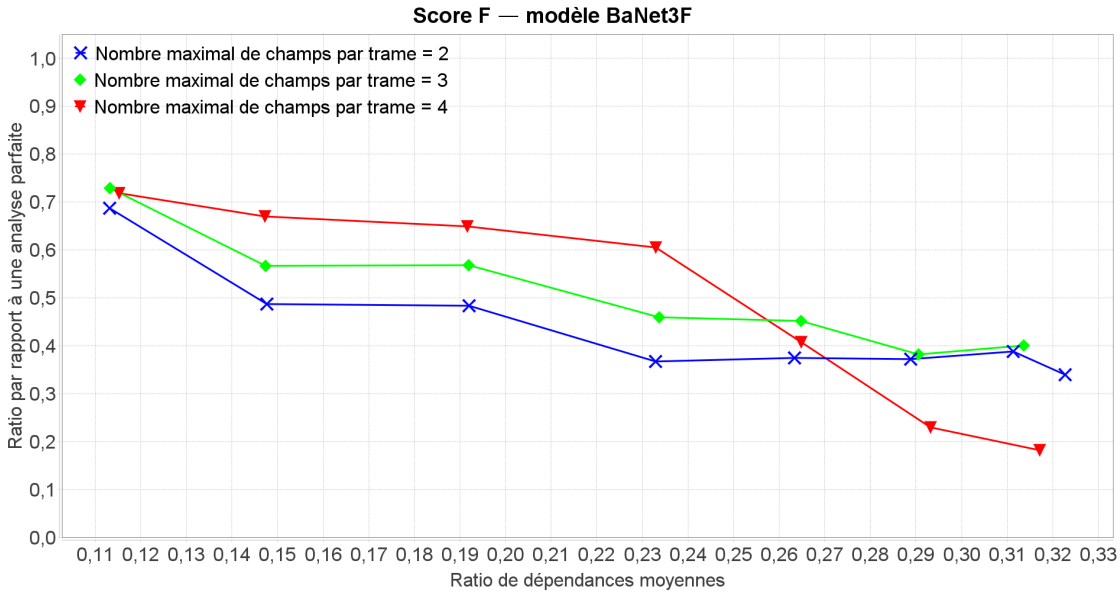


FIGURE 5.3 – Score F du modèle BaNet3F en fonction du ratio de dépendances moyennes, paramétré par le nombre maximal de champs par trame, et lissé sur 5 points

Campagne 2 : Seuil relatif de validation du découpage

Sur le graphique de la Figure 5.4, présentant le score F lissé, nous pouvons voir que la meilleure courbe présente des performances maximales s'élevant à environ 93%, pour un seuil

relatif de validation du découpage de 0.18.

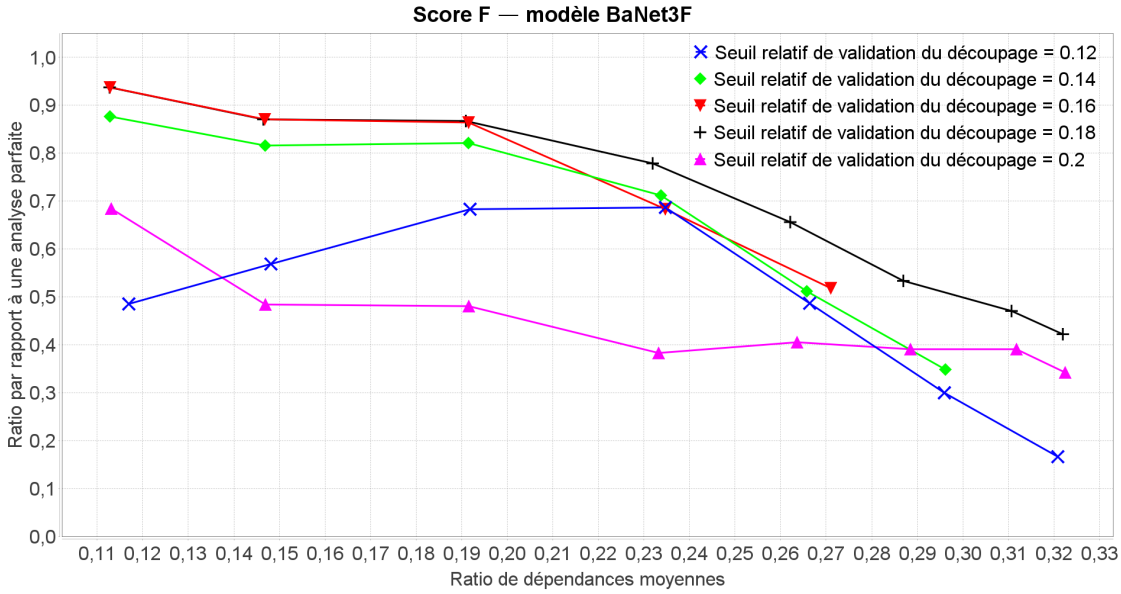


FIGURE 5.4 – Score F du modèle BaNet3F en fonction du ratio de dépendances moyennes, paramétré par le seuil relatif de validation du découpage, et lissé sur 5 points

La meilleure courbe du modèle BaNet3F est donc celle correspondant à un nombre maximal de champs par trame de 2 et un seuil relatif de validation du découpage de 0.18.

5.1.4 Comparaison des performances des modèles BaNet3F et LDA

La Table 5.3 synthétise les paramètres des meilleures courbes de performances de LDA et BaNet3F pour les traces analysées.

Le graphique de la Figure 5.5 présente le meilleur score F des modèles LDA et BaNet3F, et l'on peut constater que le modèle proposé est nettement supérieur à LDA quel que soit le ratio de dépendances moyennes considéré, c'est à dire aussi quel que soit le découpage considéré. Le modèle BaNet3F atteint des performances de 100% dans les cas les plus favorables, tandis que celles du modèle LDA se hissent seulement jusqu'aux alentours de 38%. L'écart de performances entre les deux modèles varie entre 60 et 80% dans la zone de ratio de dépendances moyennes au sein de laquelle BaNet3F performe le mieux (< 0.2), tandis qu'il se réduite à 10-30% lorsque la trace à analyser se complexifie (> 0.2).

5.2 Performances comparées des modèles BaNet3F et de Cai et al.

Nous allons maintenant comparer les performances de BaNet3F et du modèle de Cai et al. Dans un premier temps, nous expliquerons succinctement le principe de fonctionnement de ce dernier, avant de présenter la comparaison en elle-même.

Paramètre	Valeur
Nombre de trames	1000
Facteur de la distribution intra champ inital	3
Facteur de la distribution intra champs	5
Facteur de la distribution inter champs	0 ;-1 ;-2 ;-3 ;-4 ;-5
Nombre maximum de valeurs d'un champ	10
Découpage caché	4,4 3,1,1,1,2 32,32 16,8,8,32 1,2,1,3,1,8,16,32
Nombre de mots-clefs	3
Gradient maximal de probabilité	0.1
Nombre maximal de champs par trame	2
Seuil relatif de validation du découpage	0.18
Nombre de répétitions	10

TABLE 5.3 – Paramètres de la campagne de simulation comparative des modèles LDA et BaNet3F

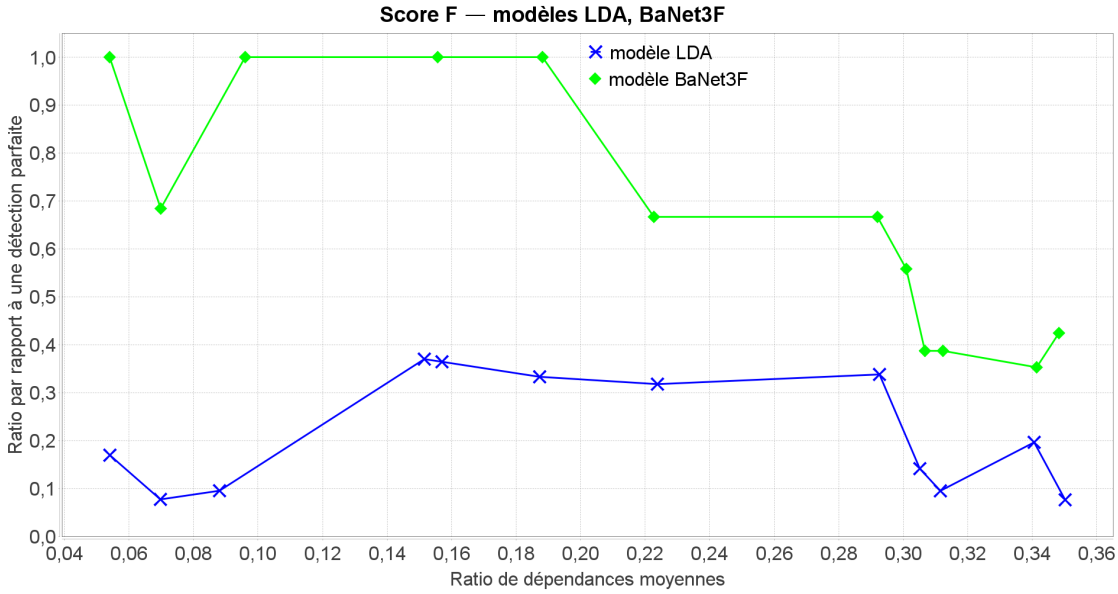


FIGURE 5.5 – Meilleurs scores F des modèles LDA et BaNet3F en fonction du ratio de dépendances moyennes

5.2.1 Modèle de Cai et al.

Le modèle proposé par Cai et al. vise, comme BaNet3F, à apprendre les formats de trames présents dans une trace. Il comporte un certain nombre d'éléments, mais nous ne présenterons

que le plus central, celui sur lequel nous nous sommes basé pour concevoir le modèle BaNet3F.

De façon très macroscopique, le modèle comporte d'abord une étape de traitement initial visant à identifier les séquences fermées récurrentes présentes dans la trace à analyser. Par séquences fermées, on entend un ensemble de séquences tel qu'aucune séquence le composant ne soit une sous-séquence d'une autre séquence de l'ensemble. Ces séquences sont concrètement des chaînes d'octets.

L'algorithme débute en considérant toutes les séquences élémentaires possibles (soit 256 × 1 octet). Ensuite, il supprime toutes les séquences dont la fréquence d'apparition est inférieure à un certain seuil, que nous appellerons le seuil de filtrage des séquences fermées récurrentes, et sauvegarde les autres. Après quoi, les séquences de N octets restantes sont superposées en décalage d'un octet, c'est à dire que l'octet 1 de l'une des séquences est aligné à l'octet 2 d'une autre, et si les parties superposées correspondent parfaitement (ont la même valeur), les deux séquences sont fusionnées, résultant en une séquence de longueur N+1. Toutes les combinaisons possibles sont considérées, et l'ensemble de nouvelles séquences de longueur N+1 sert de point de départ à l'itération suivante de l'algorithme. Le processus s'arrête lorsque plus aucune séquence ne franchit la phase de filtrage. Les séquences ayant franchi les phases de filtrage à chacune des itérations sont enfin récupérées, et toutes les séquences n'étant contenues dans aucune autre séquence sont sélectionnées comme séquences fermées récurrentes pour la suite de l'analyse par le modèle de Cai et al.

Ensuite, en son cœur, le modèle de Cai et al. comporte une chaîne de semi-Markov cachée, que nous présenterons en détail un peu plus loin. Cette chaîne est d'abord initialisée, puis ses paramètres sont ré-estimés (apprentissage), et enfin l'enchaînement d'états (champs) le plus probable pour chaque trame est inféré via l'algorithme de Viterbi.

Enfin, à l'aide des champs inférés correspondant à des mots-clés (par opposition aux champs data), les trames sont regroupées par type, et une structure type est proposée pour chaque groupe.

La procédure globale d'apprentissage des formats de trames par le modèle de Cai et al. est résumée sur le diagramme de la Figure 5.6.

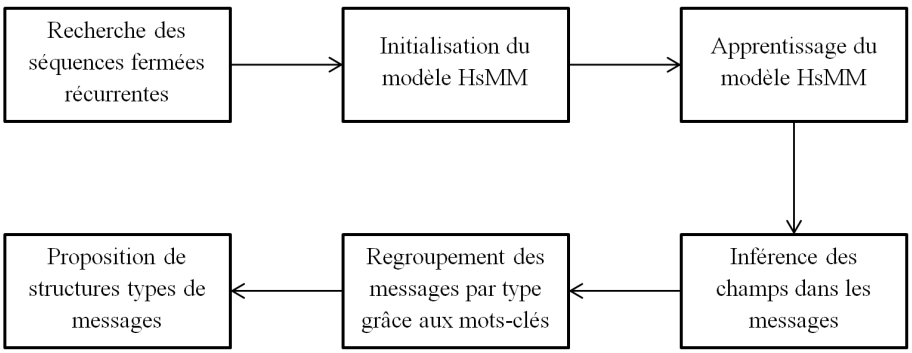


FIGURE 5.6 – Vue d'ensemble de la procédure d'apprentissage des formats de trames

Dans une chaîne de Markov cachée classique, la variable cachée est susceptible de changer à chaque instant, l'instant définissant la granularité temporelle (ou spatiale). Dans une chaîne de semi-Markov cachée, la variable cachée ne peut pas systématiquement changer d'état à chaque instant, son autorisation à changer d'état étant conditionnée par une autre variable. Cela signifie que la variable cachée est forcée à rester dans un même état pendant un multiple entier d'instants, tiré aléatoirement. On peut donc voir l'état caché comme possédant une durée variable.

Le schéma de la Figure 5.7 illustre le principe de durée variable. L'axe symbolise la suite des instants, et les flèches directement au-dessus représentent la durée d'un état de la variable cachée dans une chaîne de Markov cachée classique. Les grandes flèches au-dessus représentent une suite de durées possibles d'une chaîne de semi-Markov cachée. Les flèches sous l'axe des instants représentent les émissions d'observations à chaque instant.

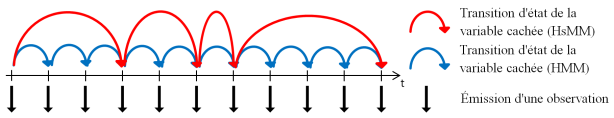


FIGURE 5.7 – Illustration de la durée variable d'un état de la variable cachée

Une chaîne de semi-Markov cachée est représentée par $\lambda = (A, B, P, \pi)$, avec A la matrice de transition de l'état caché, B la matrice d'émission des observations, P la matrice de distribution des durées des états cachés, et π la distribution initiale de l'état caché. Pour faciliter les explications par la suite, on nomme les trois variables de la chaîne de semi-Markov cachée comme suit : D, la variable représentant la durée restante de l'état caché ; S l'état caché ; et V l'observation.

Pour mieux visualiser la chaîne de semi-Markov cachée, le schéma de la Figure 5.8 la représente sous la forme d'un diagramme 2 tranches de réseau Bayésien. La chaîne de Markov étant un type de réseau Bayésien, une telle représentation s'y prête parfaitement. Les pointillés mettent en évidence les variables cachées.

Le fonctionnement de la chaîne de semi-Markov cachée est le suivant : d'abord, l'état caché est initialisé à l'aide de π, puis, à partir de l'état caché, la durée restante de cet état est tirée à l'aide de P, et une observation est générée à l'aide de B. Ensuite, à chaque instant subséquent, si la durée restante de l'état caché n'a pas atteint 0 à l'instant précédent, l'état caché conserve sa valeur, une observation est émise, et le nombre d'instants restants (durée restante) de l'état caché se décrémente de 1. Dans le cas où la durée restante de l'état caché atteint 0 à l'instant précédent, une nouvelle valeur de l'état caché est tirée à l'aide de A (potentiellement la même), une observation est émise, et une nouvelle durée d'état cachée, liée au nouvel état, est tirée.

D'un point de vue mathématique plus formel, les distributions de probabilités des différentes variables sont les suivantes :

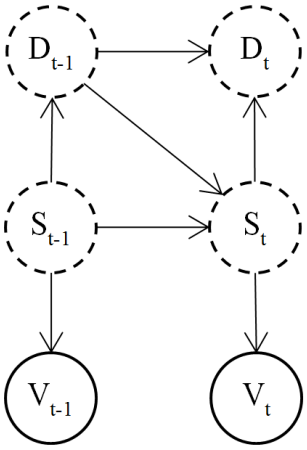


FIGURE 5.8 – Représentation sous forme graphique du modèle de génération des trames

— $P(D_t = d_t | D_{t-1} = d_{t-1}, S_t = s_t) = \begin{cases} P(d_t|s_t) = p_{s_t,d_t} \ si \ d_{t-1} = 0 \\ 1 \ si \ d_{t-1} \neq 0 \ et \ d_t = d_{t-1} - 1 \\ 0 \ sinon \end{cases}$

— $P(S_t = s_t | D_{t-1} = d_{t-1}, S_{t-1} = s_{t-1}) = \begin{cases} P(s_t|s_{t-1}) = a_{s_{t-1},s_t} \ si \ d_{t-1} = 0 \\ 1 \ si \ d_{t-1} \neq 0 \ et \ s_t = s_{t-1} \\ 0 \ sinon \end{cases}$

— $P(V_t = v_t | S_t = s_t) = b_{s_t,v_t}$

Dans le modèle de Cai et al., chaque état de la variable cachée est associé à un champ de la trame, et chaque octet appartenant à ce champ est une observation générée lorsque la chaîne se trouve dans l'état associé à ce champ. La durée de l'état représente donc la longueur du champ, et la suite des observations générées pendant la durée de cet état représente la valeur du champ.

On présente sur le schéma de la Figure 5.9 un exemple de la manière dont le modèle Cai et al. génère une trame. On peut voir que la variable cachée est initialisée à l'état i_1, dans lequel elle reste durant trois instants, émettant les octets correspondant aux lettres G, E, et T (soit le mots-clé GET, du HTTP), avant de transiter vers l'état i_2.

La variable V se voit attribuer un état pour chaque valeur d'octet possible, soit un total de 256 états de 0 à 255. La variable S se voit attribuer deux états par séquence fermée récurrente trouvée. Une moitié des états est associée aux différentes séquences fermées récurrentes, tandis que l'autre est réservée aux champs variables, dénués de valeurs caractéristiques. Enfin, la variable D se voit attribuer un nombre d'états défini par l'utilisateur. Dans l'article de présentation du modèle, ce nombre est arbitrairement fixé à 100, mais dans nos simulations, nous le fixerons à 8, car aucun des champs des traces analysées ne font plus de 8 octets.

Les matrices A et π sont initialisées aléatoirement, tandis que B et P le sont comme suit : sachant un champ associé à une séquence fermée récurrente, la probabilité qu'un octet composant cette séquence soit émis est de $e^{\frac{-l}{10}}$, où l représente la longueur de la

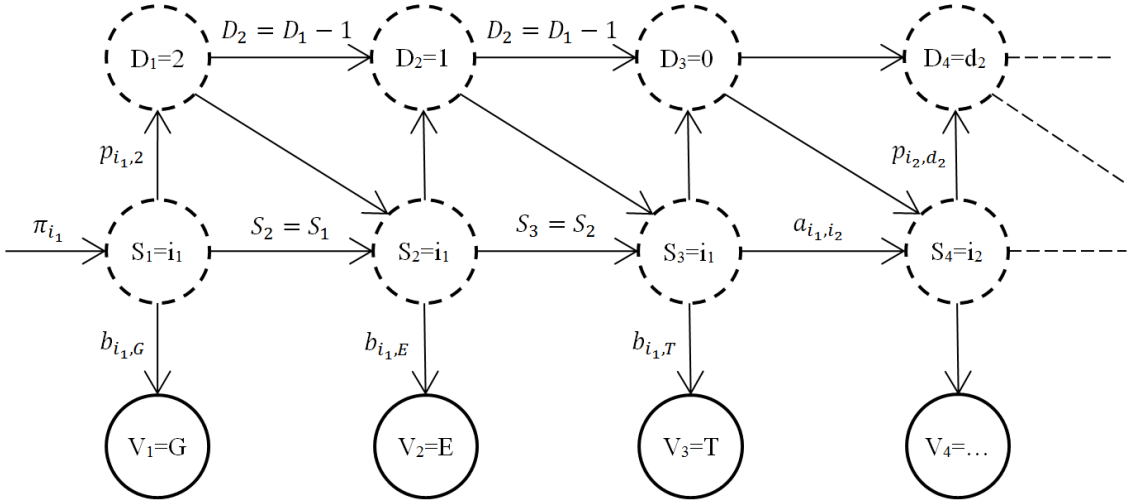


FIGURE 5.9 – Exemple d'état du modèle lors de la génération d'une trame

séquence. La probabilité qu'un octet n'appartenant pas à cette séquence soit émis est fixé à 0. Pour les champs associés à un champ variable, la probabilité d'émission d'un octet est de e^{-20}.

Quel que soit l'état de la variable cachée considérée, la probabilité que sa durée restante soit de d est initialisée à $\frac{d^2}{\sum_{k=1}^{D'} k^2}$, où D' représente le nombre total d'états de D.

Dans l'article de référence, l'apprentissage du modèle est ensuite effectué à l'aide de la méthode de Baum-Welch. Cependant, leur présentation de l'algorithme manquant de détails, et comportant certaines incohérences, nous avons décidé d'effectuer l'apprentissage à l'aide de l'algorithme EM utilisé pour notre modèle.

Ensuite, pour chaque trame, on utilise l'algorithme de Viterbi (une version spécifique dans l'article, et AMBV dans cette comparaison) pour reconstituer la succession des états et leurs durées. Cela permet de segmenter les trames en champs. La Figure 5.10 présente une exemple de résultat obtenu à l'issue de ce processus (d'après une illustration de l'article de référence).

FIGURE 5.10 – Exemple de trame HTTP segmentée en champs à l'aide de l'algorithme de Viterbi

Une fois la segmentation de toutes les trames effectuée, il est possible d'en déduire les longueurs, valeurs et positions possibles pour chacun des champs, ainsi que s'il s'agit d'un champ mot-clé ou data (grâce à l'association de certains états de la variable cachée à des mots-clés et d'autres à des champs data).

Par la suite, seules les valeurs des champs mots-clés sont utilisées, afin de regrouper les messages par types, puis proposer une structure type pour chaque groupe.

5.2.2 Cadre des comparaisons

Comme cela a été dit à plusieurs reprises, le modèle de Cai et al. est conçu pour analyser des traces au niveau de l'octet, il lui est donc impossible de trouver des champs de taille inférieure à l'octet. C'est pourquoi nous étudierons séparément le cas où les découpages cachés sont constitués à la fois de champs de longueurs inférieures et supérieures à l'octet, et le cas où les découpages sont uniquement constitués de champs de longueurs supérieures ou égales à l'octet. À noter que les mêmes campagnes sont utilisées pour chacun des deux cas, la différence se situant juste dans les post-traitements des résultats (en l'occurrence l'ajout d'un filtrage des découpages de trace sélectionnés). Les paramètres de campagne des deux parties seront donc strictement identiques.

Afin d'évaluer les performances des deux modèles, nous utiliserons le score F, calculé comme expliqué dans la section 4.1.2.

La stratégie que nous employons reste la même que celle présentée dans la section 5.1.1.

À l'aide des simulations déjà effectuées dans le chapitre 4, nous avons déterminé les paramètres dont l'étude était intéressante pour le modèle BaNet3F. Nous avons de plus constaté qu'Alpha, le paramètre de Dirichlet a priori de la distribution des mots-clés par trame, et le gradient maximal de perplexité ne semblaient pas avoir d'impact sur les performances, nous les ignorerons donc dans ce chapitre.

Pour le modèle de Cai et al., il n'existe qu'un seul paramètre :
— Le seuil de filtrage des séquences fermées récurrentes
Pour BaNet3F, les paramètres intéressants sont :
— Le seuil relatif de validation du découpage
— Le nombre maximal de champs par trame

5.2.3 Performances du modèle de Cai et al. pour des champs de toutes longueurs

La Table 5.4 synthétise les paramètres des campagnes de simulations visant à déterminer la meilleure courbe de performances du modèle de Cai et al. pour les traces comportant des champs de longueurs à la fois inférieures et supérieures à l'octet.

Pour les valeurs du seuil de filtrage des séquences fermées récurrentes utilisées, nous sommes partis des valeurs employées pour le seuil relatif de validation du découpage dans les simulations du chapitre 4, et avons a posteriori sélectionné les plus intéressantes.

Chaque point des courbes de performances est calculé en moyennant sur tous les découpages cachés différents, et en effectuant un lissage identique à celui employé dans le chapitre 4.

Paramètre	Valeur
Nombre de trames	1000
Facteur de la distribution intra champ inital	3
Facteur de la distribution intra champs	5
Facteur de la distribution inter champs	0 ;-1 ;-2 ;-3 ;-4 ;-5
Nombre maximum de valeurs d'un champ	10
Découpage caché	4,4 3,1,1,1,2 32,32 16,8,8,32 1,2,1,3,1,8,16,32
Seuil de filtrage des séquences fermées récurrentes	0.1 ;0.12 ;0.14 ;0.16 ;0.18
Nombre de répétitions	10

TABLE 5.4 – Paramètres de la campagne de simulation du modèle de Cai et al. pour tous les découpages

Seuil de filtrage des séquences fermées récurrentes

On peut voir sur le graphique de la Figure 5.11, présentant le score F lissé, que les meilleures performances du modèle de Cai et al. s'élèvent à environ 34%, et sont atteintes pour un seuil de filtrage des séquences fermées récurrentes de 0.16.

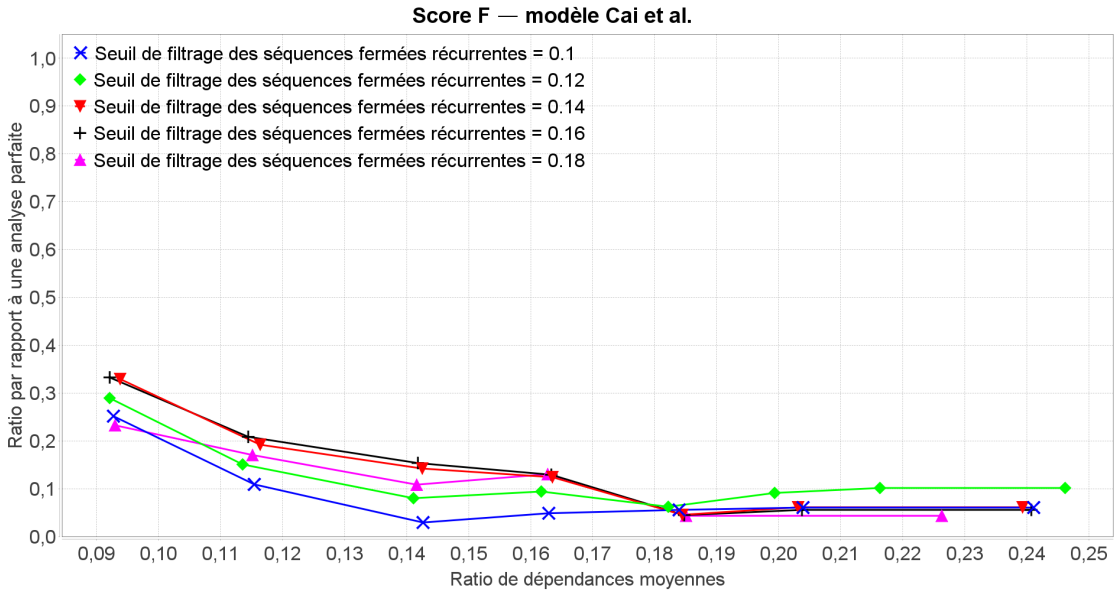


FIGURE 5.11 – Score F du modèle Cai et al. lorsqu'appliqué à tous les découpages, en fonction du ratio de dépendances moyennes, paramétré par le seuil de filtrage des séquences fermées récurrentes, et lissé sur 5 points

La meilleure courbe du modèle de Cai et al. est donc celle pour un seuil de filtrage des séquences fermées récurrentes de 0.16.

5.2.4 Performances du modèle BaNet3F pour des champs de toutes longueurs

La Table 5.5 synthétise les paramètres des campagnes de simulations visant à déterminer la meilleure courbe de performances du modèle BaNet3F pour les traces comportant des champs de longueurs à la fois inférieures et supérieures à l'octet.

Paramètre	**Campagne 1**	**Campagne 2**
Nombre de trames	1000	1000
Facteur de la distribution intra champ inital	3	3
Facteur de la distribution intra champs	5	5
Facteur de la distribution inter champs	0 ;-1 ;-2 ;-3 ;-4 ;-5	0 ;-1 ;-2 ;-3 ;-4 ;-5
Nombre maximum de valeurs d'un champ	10	10
Découpage caché	4,4 3,1,1,1,2 32,32 16,8,8,32 1,2,1,3,1,8,16,32	4,4 3,1,1,1,2 32,32 16,8,8,32 1,2,1,3,1,8,16,32
Nombre maximal de champs par trame	2 ;3 ;4 ;5 ;6	2
Seuil relatif de validation du découpage	0.2	0.12 ;0.14 ;0.16 ;0.18 ;0.2
Nombre de répétitions	10	10

TABLE 5.5 – Paramètres des campagnes de simulation du modèle BaNet3F pour tous les découpages

Pour les valeurs du nombre maximal de champs par trame et du seuil relatif de validation du découpage utilisées, nous sommes partis des valeurs employées pour la comparaison avec le modèle LDA, que nous avons ajusté a posteriori le cas échéant.

Chaque point des courbes de performances est calculé en moyennant sur tous les découpages cachés différents, et en effectuant un lissage identique à celui employé dans le chapitre 4.

Campagne 1 : Nombre maximal de champs par trame

Sur le graphique de la Figure 5.12 représentant le score F lissé, on constate que les meilleures performances montent à environ 83%, pour un nombre maximal de champs par trame de 4.

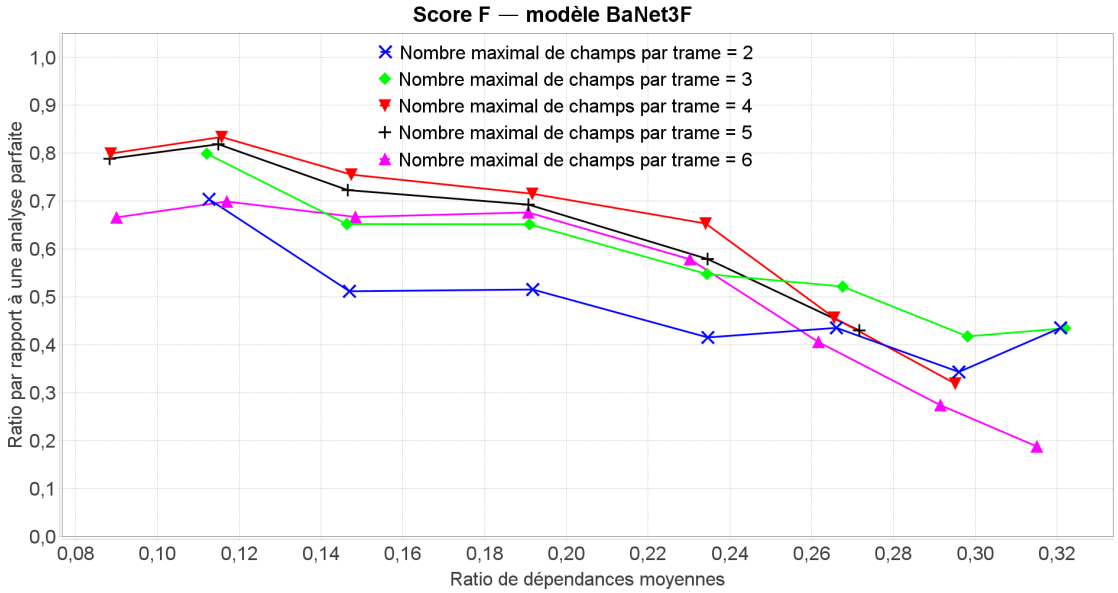


FIGURE 5.12 – Score F du modèle BaNet3F lorsqu'appliqué à tous les découpages, en fonction du ratio de dépendances moyennes, paramétré par le nombre maximal de champs par trame, et lissé sur 5 points

Campagne 2 : Seuil relatif de validation du découpage

Le graphique de la Figure 5.13, présentant le score F lissé, nous montre que les meilleures performances s'élèvent à environ 93%, pour un seuil relatif de validation du découpage de 0.18.

La meilleure courbe est donc celle pour un nombre maximal de champs par trame de 2 et un seuil relatif de validation du découpage de 0.18.

5.2.5 Comparaison des performances des modèles BaNet3F et de Cai et al. pour des champs de toutes longueurs

La Table 5.6 synthétise les paramètres des meilleures courbes de performances du modèle de Cai et al et BaNet3F pour les traces comportant des champs de longueurs à la fois inférieures et supérieures à l'octet.

Sur le graphique de la Figure 5.14 présentant le meilleur score F du modèle de Cai et al. et du modèle BaNet3F, on peut constater que le modèle proposé est nettement supérieur à celui de Cai et al. quel que soit le ratio de dépendances moyennes considéré, c'est à dire aussi quel que soit le découpage considéré. Le modèle BaNet3F atteint des performances de 100% dans les cas les plus favorables, tandis que celles du modèle de Cai et al. se hissent seulement jusqu'aux alentours de 62% dans de rares cas. De plus, les performances de ce dernier chutent après 0.12 de ratio de dépendances moyennes, devenant presque nulles. L'écart de performances entre les deux modèles varie entre 40 et 100% dans la zone de ratio de dépendances moyennes au sein de laquelle BaNet3F performe le mieux (RDM < 0.2), tandis qu'il se réduit à 30-70% lorsque la trace à analyser se complexifie (RDM > 0.2).

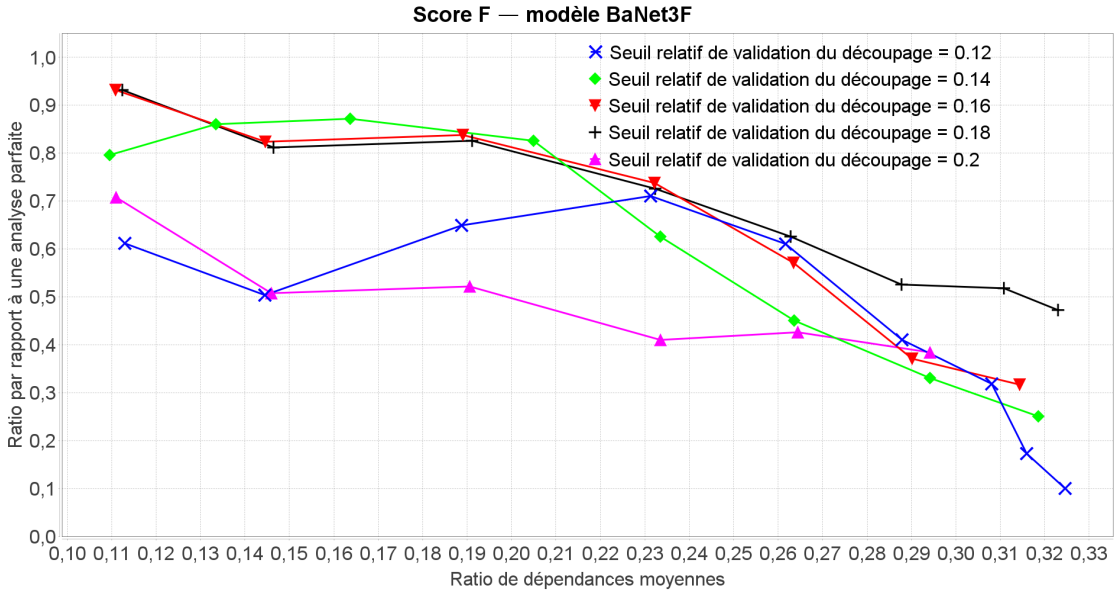


FIGURE 5.13 – Score F du modèle BaNet3F lorsqu'appliqué à tous les découpages, en fonction du ratio de dépendances moyennes, paramétré par le seuil relatif de validation du découpage, et lissé sur 5 points

Paramètre	Valeur
Nombre de trames	1000
Facteur de la distribution intra champ inital	3
Facteur de la distribution intra champs	5
Facteur de la distribution inter champs	0 ;-1 ;-2 ;-3 ;-4 ;-5
Nombre maximum de valeurs d'un champ	10
Découpage caché	4,4 3,1,1,1,2 32,32 16,8,8,32 1,2,1,3,1,8,16,32
Seuil de filtrage des séquences fermées récurrentes	0.16
Nombre maximal de champs par trame	2
Seuil relatif de validation du découpage	0.18
Nombre de répétitions	10

TABLE 5.6 – Paramètres de la campagnes de simulation comparative des modèles de Cai et al. et BaNet3F pour tous les découpages

5.2.6 Performances du modèle de Cai et al. pour des découpages en octets uniquement

La Table 5.7 synthétise les paramètres des campagnes de simulations visant à déterminer la meilleure courbe de performances du modèle de Cai et al. pour les traces comportant

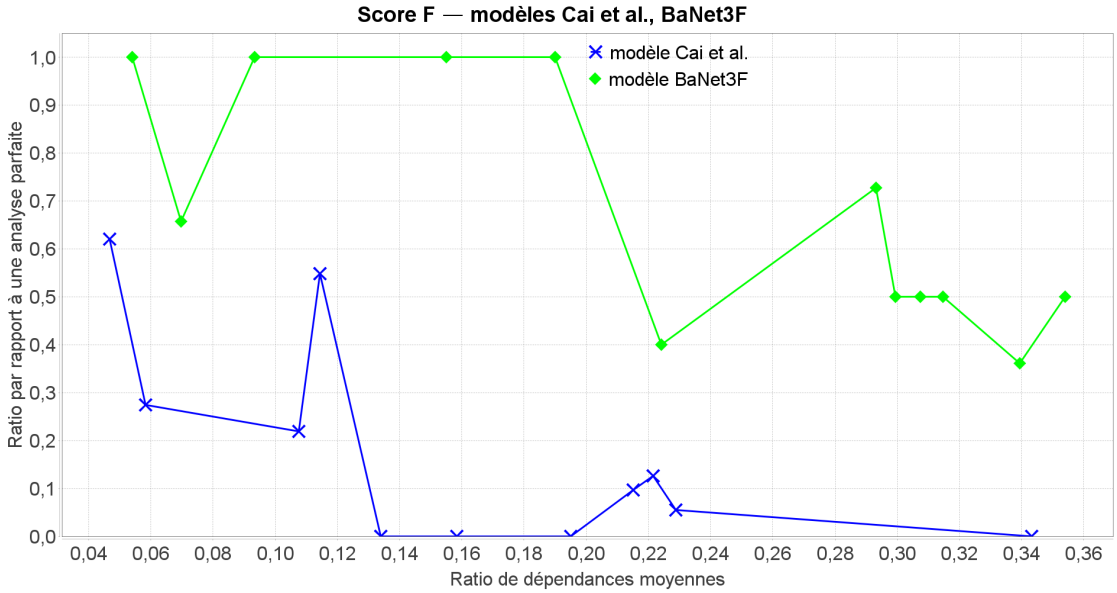


FIGURE 5.14 – Meilleurs scores F des modèles Cai et al. et BaNet3F lorsqu'appliqués à tous les découpages, en fonction du ratio de dépendances moyennes

uniquement des champs de longueurs supérieures ou égales à l'octet.

Paramètre	Valeur
Nombre de trames	1000
Facteur de la distribution intra champ inital	3
Facteur de la distribution intra champs	5
Facteur de la distribution inter champs	0 ;-1 ;-2 ;-3 ;-4 ;-5
Nombre maximum de valeurs d'un champ	10
Découpage caché	32,32 16,8,8,32
Seuil de filtrage des séquences fermées récurrentes	0.1 ;0.12 ;0.14 ;0.16 ;0.18
Nombre de répétitions	10

TABLE 5.7 – Paramètres de la campagne de simulation du modèle de Cai et al. pour les découpages en octets

Seuil de filtrage des séquences fermées récurrentes

On peut voir sur le graphique de la Figure 5.15, présentant le score F lissé, que les courbes pour un seuil de filtrage des séquences fermées récurrentes de 0.14 et 0.16 sont approximativement équivalentes en moyenne sur la fenêtre de ratio de dépendances moyennes

considérée. Nous privilégieront cependant la valeur 0.16, car trois des quatre points la constituant sont supérieures à l'alternative. Ainsi, les plus fortes performances atteintes par le modèle de Cai et al. sont d'environ 62%.

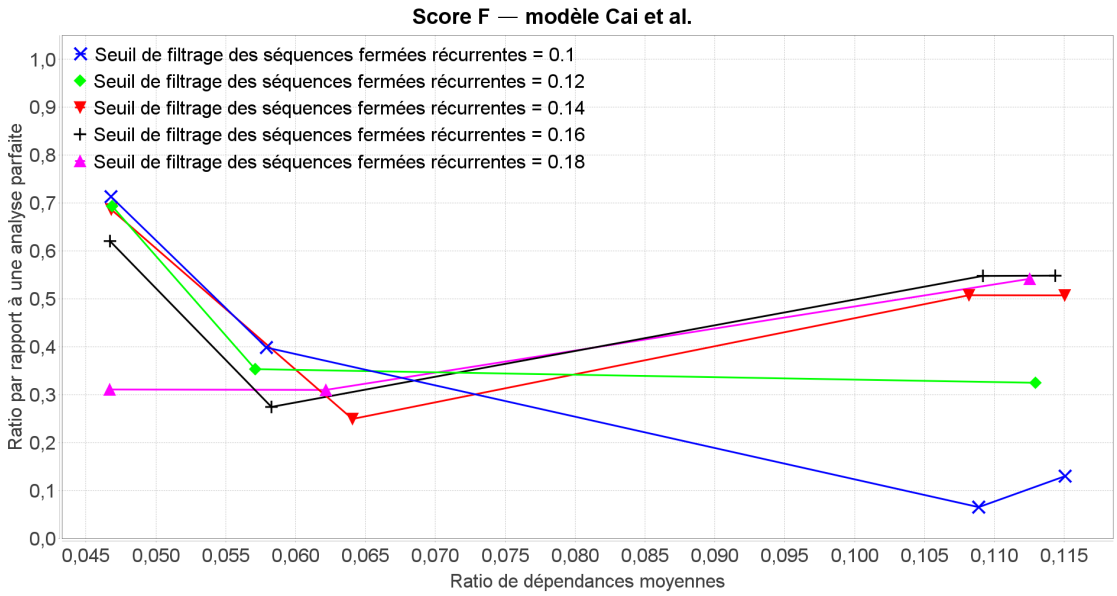


FIGURE 5.15 – Score F du modèle Cai et al. lorsqu'appliqué aux découpages en octets, en fonction du ratio de dépendances moyennes, paramétré par le seuil de filtrage des séquences fermées récurrentes

La meilleure courbe est donc celle pour un seuil de filtrage des séquences fermées récurrentes de 0.16.

5.2.7 Performances du modèle BaNet3F pour des découpages en octets uniquement

La Table 5.8 synthétise les paramètres des campagnes de simulations visant à déterminer la meilleure courbe de performances de BaNet3F pour les traces comportant uniquement des champs de longueurs supérieures ou égales à l'octet.

Campagne 1 : Nombre maximal de champs par trame

Sur le graphique de la Figure 5.16 présentant le score F lissé, on constate que les meilleures performances atteintes par BaNet3F sont de 100%, pour un nombre maximal de champs par trame de 2.

Campagne 2 : Seuil relatif de validation du découpage

On observe sur le graphique de la Figure 5.17, présentant le score F lissé, que les meilleures performances sont également de 100%, pour des seuils de validation du découpage

Paramètre	Campagne 1	Campagne 2
Nombre de trames	1000	1000
Facteur de la distribution intra champ inital	3	3
Facteur de la distribution intra champs	5	5
Facteur de la distribution inter champs	0 ;-1 ;-2 ;-3 ;-4 ;-5	0 ;-1 ;-2 ;-3 ;-4 ;-5
Nombre maximum de valeurs d'un champ	10	10
Découpage caché	32,32 16,8,8,32	32,32 16,8,8,32
Nombre maximal de champs par trame	2 ;3 ;4 ;5 ;6	2
Seuil relatif de validation du découpage	0.2	0.12 ;0.14 ;0.16 ;0.18 ;0.2
Nombre de répétitions	10	10

TABLE 5.8 – Paramètres des campagnes de simulation du modèle BaNet3F pour les découpages en octets

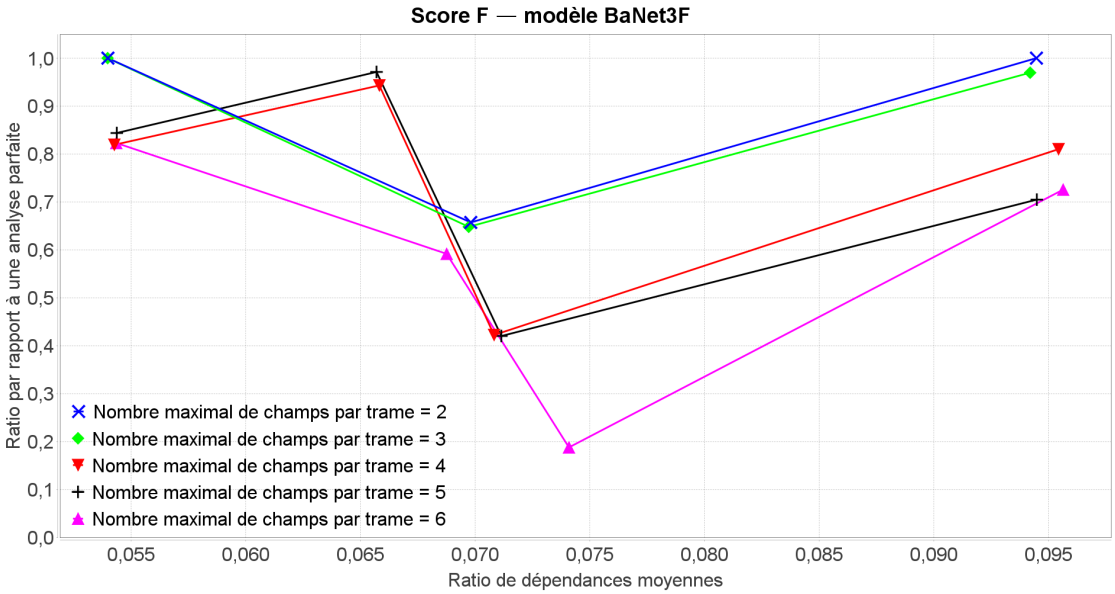


FIGURE 5.16 – Score F du modèle BaNet3F lorsqu'appliqué aux découpages en octets, en fonction du ratio de dépendances moyennes, paramétré par le nombre maximal de champs par trame

de 0.16,0.18, et 0.2.

Toutes les meilleures courbes atteignent les 100% de performances et sont par ailleurs très proches, il n'y en a donc pas de meilleure que les autres, mais puisque nous devons en sélectionner une, nous avons pris celle correspondant à un nombre maximal de champs par trame de 2 et un seuil relatif de validation du découpage de 0.2.

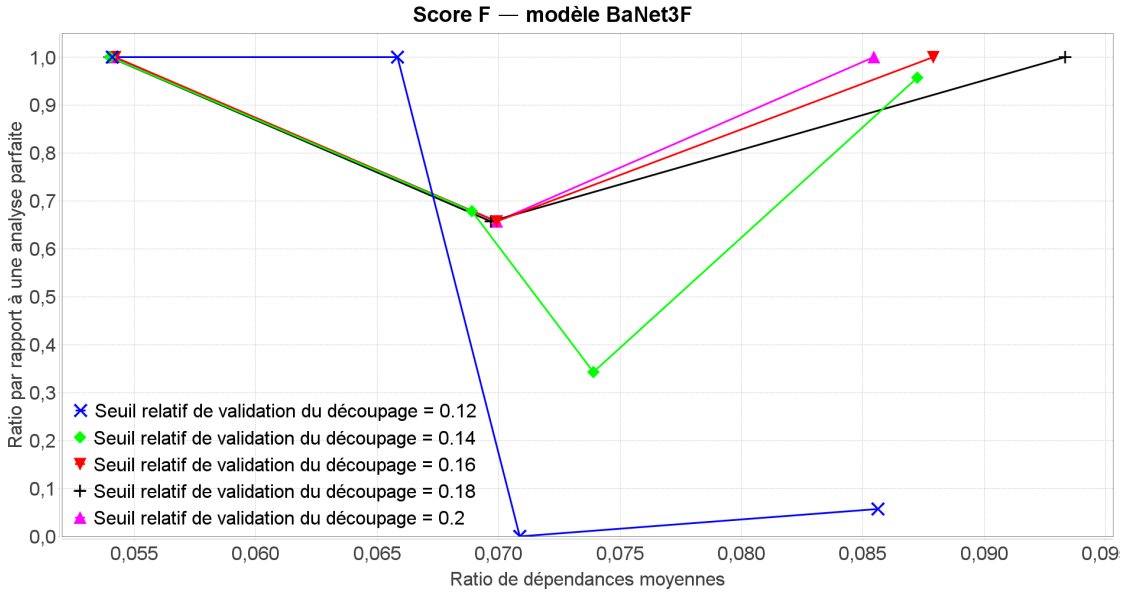


FIGURE 5.17 – Score F du modèle BaNet3F lorsqu'appliqué aux découpages en octets, en fonction du ratio de dépendances moyennes, paramétré par le seuil relatif de validation du découpage

5.2.8 Comparaison des performances des modèles BaNet3F et de Cai et al. pour des découpages en octets uniquement

La Table 5.9 synthétise les paramètres des meilleures courbes de performances du modèle de Cai et al et BaNet3F pour les traces comportant uniquement des champs de longueurs supérieures ou égales à l'octet.

Le graphique de la Figure 5.18, présentant le score F lissé, montre clairement que même lorsque seuls les champs de longueurs supérieures ou égales à l'octet sont considérés, c'est à dire lorsque l'on se rapproche autant que possible (étant donné la nature binaire du protocole) du domaine pour lequel a été conçu le modèle de Cai et al., les performances de BaNet3F restent largement supérieures. Cependant, les performances du modèle de Cai et al. sont nettement meilleures en moyenne qu'elles ne l'étaient lorsque tous les découpages étaient considérés, même si les meilleures performances restent identiques (62%), car elles correspondent à un découpage particulier présent dans les deux cas de simulation. Les performances de BaNet3F sont également meilleures en moyenne que dans le cas précédent, si bien que l'écart entre les deux modèles varie entre 30% et 50%. On peut toutefois noter que l'écart s'est nettement réduit par rapport à ce qu'il était lorsque tous les découpages étaient considérés.

5.3 Conclusion

Nous pouvons conclure de cette comparaison de notre modèle BaNet3F avec les modèles LDA et de Cai et al. que, quel que soit le ratio de dépendances moyennes ou le découpage consi-

Paramètre	Valeur
Nombre de trames	1000
Facteur de la distribution intra champ inital	3
Facteur de la distribution intra champs	5
Facteur de la distribution inter champs	0 ;-1 ;-2 ;-3 ;-4 ;-5
Nombre maximum de valeurs d'un champ	10
Découpage caché	32,32 16,8,8,32
Seuil de filtrage des séquences fermées récurrentes	0.16
Nombre maximal de champs par trame	2
Seuil relatif de validation du découpage	0.2
Nombre de répétitions	10

TABLE 5.9 – Paramètres de la campagne de simulation comparative des modèles de Cai et al. et BaNet3F pour les découpages en octets

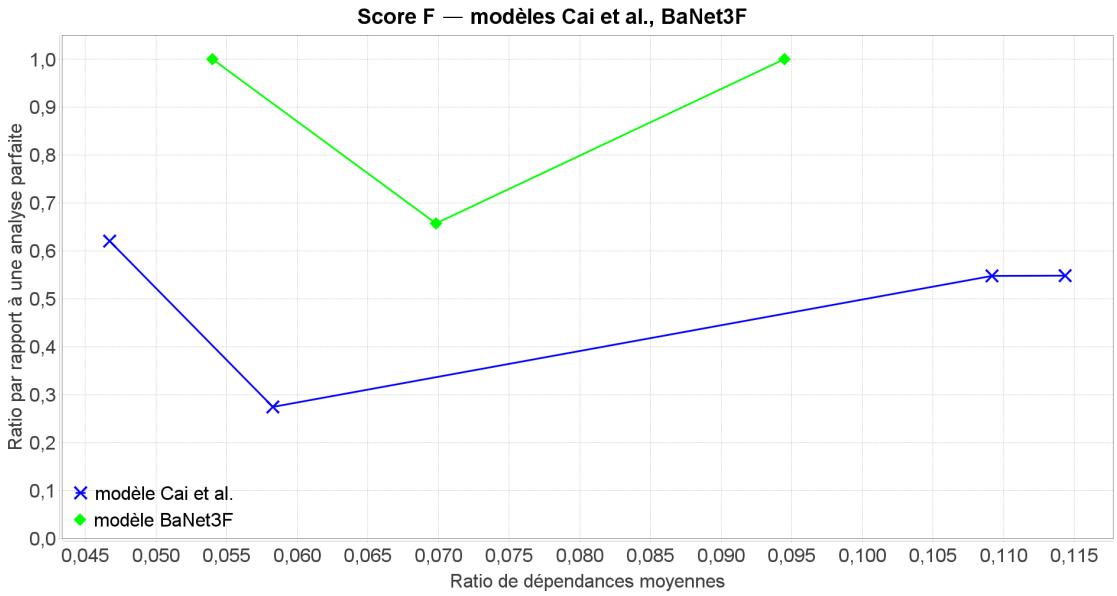


FIGURE 5.18 – Meilleurs scores F des modèles Cai et al. et BaNet3F lorsqu'appliqués aux découpages en octets, en fonction du ratio de dépendances moyennes

déré, BaNet3F est au-dessus des deux autres modèles. Les performances du modèle BaNet3F atteignent même parfois 100% dans les cas les plus favorables (faible ratio de dépendances moyennes et découpage simples). L'écart a également tendance a se réduire lorsque le ratio de dépendances moyennes augmente, passant de plus de 50% de différence en terme de score F,

à moins de 50%.

Conclusion générale et perspectives

L'objectif de cette thèse était d'étudier un moyen de permettre à un object communicant placé dans un environnement protocolaire inconnu, d'établir la communication avec les objets du ou des réseaux à portée radio.

À cette fin, nous avons adopté une démarche en deux étapes principales :

1. Un apprentissage des formats de trames employés par les protocoles de la pile protocolaire inconnue, ainsi que leurs machines d'états.
2. L'affectation d'une signification à chacun des états et formats précédemment dégagés à partir de connaissances possédées a priori.

Compte tenu du caractère très ambitieux de cet objectif, nous nous sommes limités à la première partie de la démarche, et plus précisément à l'apprentissage des formats de trames du protocole de la couche Liaison de Données. Nous avons également posé l'hypothèse simplificatrice de la présence d'un format de trame unique dans la trace à analyser.

Dans un premier temps, nous avons étudié théoriquement trois algorithmes d'identification de séquences utilisés dans des modèles de Protocole Reverse Engineering visant à apprendre les formats de trame de protocoles inconnus : Variance Distribution of the Variances (VDV) [57] issu du modèle proposé par Trifilò et al., Aho-Corasick modifié (AC) [58] issu du modèle proposé par Wang et al., et Latent Dirichlet Allocation (LDA) [59] issu du modèle ProDecoder proposé par Wang et al. Nous avons ainsi pu voir qu'il existait plusieurs méthodes pour effectuer de l'identification de séquence, allant des plus basiques, employant de simples propriétés statistiques (VDV), au plus complexes, usant de théorie probabiliste poussée (LDA).

Suite à cette étude théorique, nous avons effectué une simulation comparative des trois algorithmes appliqués à des traces Liaison de Données aléatoires générées selon la norme Zigbee [64] par un générateur conçu par nos soins. Les résultats de cette comparaison nous ont clairement montré que les performances de l'algorithme LDA étaient nettement supérieures à celles des deux autres. En effet, suivant les métriques considérées, les performances de LDA variaient entre 50% et 80%, tandis que celles d'AC oscillaient entre 40% et 50%, et celles de VDV entre 10% et 40%. L'algorithme LDA étant issu du domaine des réseaux Bayésiens, nous avons décidé de nous orienter sur cette voie.

Nous avons ensuite présenté les bases des réseaux Bayésiens, à savoir leurs représentations graphique et mathématique, les notions d'inférence et d'apprentissage, avec une technique permettant de les réaliser [67]. En s'appuyant sur ces notions, nous avons introduit et

expliqué le fonctionnement du modèle Bayesian Network Frame Format Finder (BaNet3F) proposé dans cette thèse. De plus, dans le cadre du développement de ce modèle, nous avons conçu une variante de l'algorithme de Viterbi, Automatic Markov Boundaries construction optimized Viterbi (AMBV), permettant de trouver l'état le plus probable d'un réseau Bayésien quelconque de façon relativement optimisée.

Une fois le modèle présenté, nous avons testé ses performances sur des traces générées ainsi que sur des traces réelles. Les premières étaient issues d'un générateur basé sur un réseau Bayésien, que nous avions nous-même créé afin de maîtriser la composition statistique des traces. La longueur ainsi que le contenu des champs ont été déterminés de façon purement arbitraire de notre part. Les résultats nous ont montré que lorsque la structure des trames reste simple (peu de champs et peu de dépendance entre les champs), les résultats sont très bons (>80% de champs correctement détectés) voire dans certains cas spécifiques parfaits (100% de champs correctement détectés). Les traces réelles utilisées dans les simulations étaient issues de traces du protocole IEEE 802.15.4 pour la couche Liaison de Données. Suite aux simulations, nous avons constaté que les performances sur traces réelles étaient globalement plus faibles que sur traces générées, ne dépassant guère les 50% dans les meilleurs cas de figure. Les principales explications de cette baisse de performance que nous pouvons avancer sont que les traces réelles contiennent généralement beaucoup plus de dépendance, ainsi que des champs à valeur fixe.

Enfin, nous avons comparé BaNet3F à deux modèles de l'état de l'art : le modèle LDA, qui s'était révélé le plus performant des trois comparés précédemment, et le modèle de Cai et al. [60], qui nous a servi de point de départ pour concevoir notre propre modèle, BaNet3F. Les résultats des simulations nous ont permis d'affirmer que notre modèle est nettement plus performant que les deux issus de l'état de l'art. Nous avons observé un écart de plus de 50% lorsque la complexité du découpage est faible, même si cet écart a tendance à se réduire lorsque la complexité augmente.

Nous espérions initialement obtenir de très bonnes performances sur traces réelles, semblables à celles obtenues sur traces générées de faible complexité. Cela nous aurait permis de poser une base solide sur laquelle s'appuyer pour avancer dans la démarche globale présentée un peu plus haut. Bien qu'insuffisants pour considérer l'apprentissage d'un format de trame de couche DLL comme fiable, les résultats actuels sont cependant encourageants. Afin de les améliorer, nous avons pensé à un certain nombre d'axes de recherches à explorer à la suite du travail déjà effectué.

Nous savons que les performances sont élevées lorsque la dépendance entre les champs est faible ; il serait donc intéressant de creuser ce point afin d'introduire dans le réseau un mécanisme de prise en compte de cette dépendance plus poussé que notre simple variable ValeurChamp.

Nous avons également constaté que la présence de valeurs de champs fixes entraîne une baisse des performances ; se pencher sur un mécanisme à intégrer dans le modèle pour pallier ce phénomène constitue une autre voie d'amélioration.

La métrique utilisée dans le cadre de la validation du découpage est relativement simpliste ;

il pourrait donc s'avérer intéressant de la retravailler.

Afin de limiter la complexité de notre réseau, nous avons mis en place une procédure de simplification automatique des traces à analyser, perdant ainsi une importante partie de l'information. Trouver une façon de garder la maîtrise de la complexité sans pour autant devoir procéder à cette simplification, ou alors la limiter nettement plus que nous ne l'avons fait, serait l'idéal. Une autre possibilité serait de réintroduire l'information supprimée au fur et à mesure des découpages successifs des traces par le modèle. En effet, ces découpages simplifient naturellement les analyses effectuées à chaque itération, il deviendrait donc possible d'y réintroduire progressivement de la complexité tout en maîtrisant la complexité globale de chaque analyse.

Dans les différentes campagnes de simulations du modèle BaNet3F, nous avons tenté une optimisation manuelle des hyperparamètres du modèle, mais il pourrait être intéressant d'adjoindre une couche supplémentaire d'apprentissage au modèle afin qu'il apprenne lui-même ses hyperparamètres, à partir d'un nombre de paramètres d'optimisation inférieur.

Enfin, nous avons envisagé une dernière piste, consistant à emprunter une voie complètement différente de celle choisie dans cette thèse, à savoir employer les réseaux de neurones au lieu des réseaux Bayésiens. Il serait intéressant de comparer ces deux approches, tant en matière d'efficacité que d'efficience.

Le schéma de la Figure 5.19 présente une vue d'ensemble décomposée de la démarche globale introduite en début de thèse. Cela permet de mettre en perspective ce qui a été étudié dans cette thèse (encadré en pointillés) par rapport à l'ampleur totale de la tâche d'établissement automatique de la communication avec un réseau inconnu. Ainsi, on peut voir que plusieurs axes d'amélioration restent à explorer dans le cadre de la problématique initiale de la thèse.

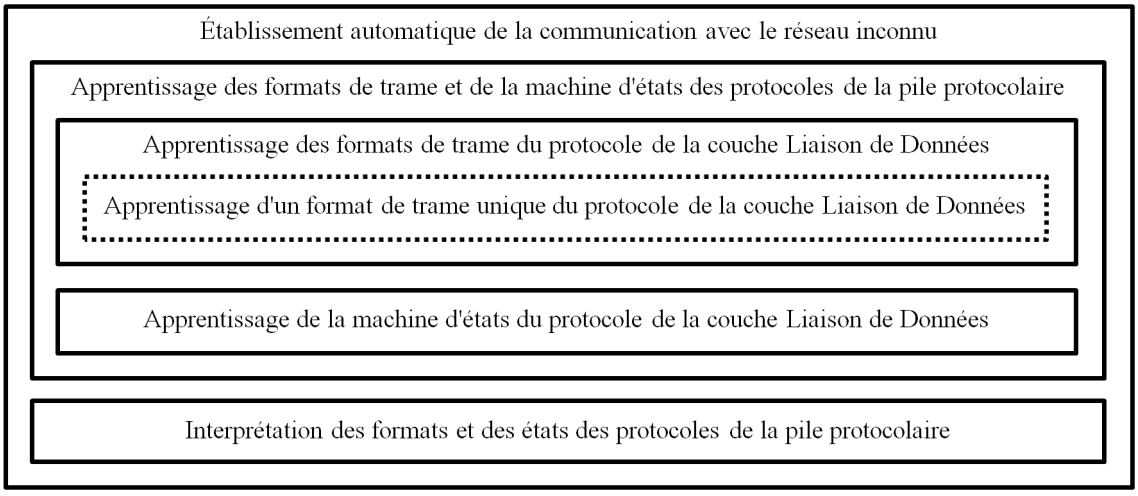


FIGURE 5.19 – Mise en perspective du travail effectué dans le cadre de la thèse par rapport à la démarche globale proposée

Nous avions mentionné dans l'introduction, section 2.2 que les organismes à l'origine des protocoles Zigbee et Thread étaient en train de s'accorder pour créer une norme commune s'inspirant de leurs deux protocoles. Cette information datait de début 2019, et depuis, la Zigbee Alliance ainsi que le Thread Group ont revu leurs ambitions à la hausse, fédérant autour d'eux nombre de grandes entreprises, et annonçant fin 2019 la création d'un nouveau protocole : le Connected Home over IP ou CHIP [73]. Durant la rédaction du présent manuscrit, l'arrivée du nouveau protocole a été officiellement annoncée, sous le nom de Matter, et la Zigbee Alliance a été renommée Connectivity Standards Alliance [74]. Celle-ci a pour mission de gérer le standard sous-jacent à Matter ainsi que sa compatibilité avec les différents standards domotiques comme Zigbee, Thread, HomeKit, ou Google Home. Ainsi, avec 180 grandes entreprises au niveau mondial derrière ce nouveau standard et son interconnectabilité annoncée avec tous les protocoles domotiques répandus, il semble que Matter est destiné à s'imposer comme standard de référence dans les années à venir.

Bien que la problématique initiale de cette thèse présente à ce jour un intérêt moindre, il est cependant possible d'utiliser le modèle proposé BaNet3F pour d'autres usages. En effet, il peut servir à apprendre le format de trames présent au sein d'une trace à d'autres fins que pour établir la communication avec un réseau IoT. Par exemple, en matière de sécurité, pour analyser un protocole inconnu dont on cherche à déterminer la dangerosité et à se protéger, ou alors pour évaluer les vulnérabilités d'un protocole que l'on compte utiliser, sans avoir accès au code source ou à sa documentation. De plus, l'usage de notre modèle ne se limite pas à l'analyse de traces réseaux, bien qu'il ait été conçu pour ; il pourrait potentiellement être appliqué à la découverte de structures cachées dans n'importe quelle séquence de longueur finie constituée d'éléments pris dans un alphabet fini (et relativement restreint, de préférence). Par exemple, il serait éventuellement possible de l'appliquer à la reconnaissance de parole. Par ailleurs, l'algorithme AMBV que nous avons conçu peut être appliqué dans n'importe quel domaine faisant usage des réseaux Bayésiens. Ainsi, BaNet3F pourrait servir de base de réflexion pour répondre à la problématique d'un sujet dans un autre domaine.

Annexe 1 : Loi de Dirichlet

La loi de Dirichlet de dimension K≥2 de paramètre $\alpha = \{\alpha_k\}_{k=1}^K$ et générant un vecteur $\{x_k\}_{k=1}^K$, possède pour densité de probabilité :

$$f(\{x_k\}_{k=1}^K, \{\alpha_k\}_{k=1}^K) = \frac{\Gamma(\sum_{i=1}^K \alpha_i)}{\prod_{i=1}^K \Gamma(\alpha_i)} \prod_{i=1}^K x_i^{\alpha_i - 1} \tag{5.1}$$

, avec $\forall k$, $0 < x_k < 1$ et $\sum_{i=1}^K x_i = 1$, et $\alpha_k > 0$

Annexe 2 : Chaînes de Markov

Chaînes de Markov simples

Les Chaînes de Markov ou Modèles de Markov sont une famille de processus visant à modéliser le comportement d'un système. Elles se présentent sous la forme d'un ensemble dénombrable d'états possibles du système, liés les uns aux autres par des probabilités de transition d'un état à l'autre. On note la variable représentant l'état du système X, et l'espace de ses états possibles E.

La propriété fondamentale d'une Chaîne de Markov est que l'état du système à un instant donné ne dépend que de son état à l'instant précédent. L'état de la chaîne à un instant t donné est représenté par le vecteur $vecx_t$ explicitant la probabilité de se trouver dans chaque état possible de X.

Une Chaîne de Markov est caractérisée par deux entités :
— $\vec{\nu}$, son vecteur d'initialisation exprimé en termes de probabilités de X de se trouver dans chaque état de E à l'instant initial.
— Π, sa matrice de transition exprimée en termes de probabilités pour X de passer à l'état suivant, quel qu'il soit, sachant l'état actuel, quel qu'il soit.

On passe d'une distribution d'états à la suivante par la formule $vecx_t = \Pi \times vecx_{t-1}$. La distribution des états à un instant donné n quelconque s'exprime par $vecx_n = \prod_{i=1}^{n} [\Pi] \times \vec{\nu}$.

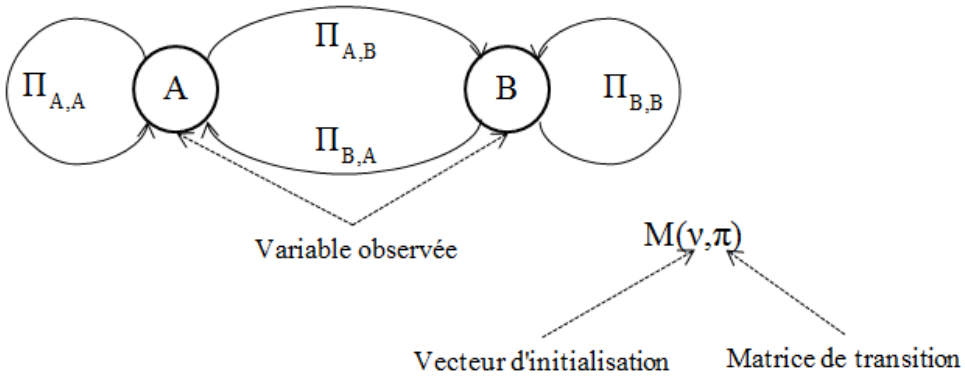
Un exemple d'une Chaîne de Markov à deux états A et B est présenté sur la Figure 5.20.

Chaînes de Markov Cachées

Les Chaînes de Markov Cachées se basent sur des Chaînes de Markov simples dont on considère l'état comme caché.

Les variables observées sont des émissions Y générées par les états X. L'état des observations à un instant t donné est représenté par le vecteur $vecy_t$ explicitant la probabilité d'observer chaque état Y. On note l'espace des observations K.

Une Chaîne de Markov Cachée est caractérisée par trois entités :



$$X \in \{A,B\}$$

$$\nu = \begin{bmatrix} \nu_A \\ \nu_B \end{bmatrix} \quad \begin{matrix} \nu_A = P(X_0 = A) \\ \nu_B = P(X_0 = B) \end{matrix} \qquad x_n = \begin{bmatrix} X_A \\ X_B \end{bmatrix} \quad \begin{matrix} X_A = P(X_n = A) \\ X_B = P(X_n = B) \end{matrix}$$

$$\Pi = \begin{bmatrix} \Pi_{A,A} & \Pi_{A,B} \\ \Pi_{B,A} & \Pi_{B,B} \end{bmatrix} \quad \begin{matrix} \Pi_{A,A} = P(X_{n+1} = A | X_n = A) \\ \Pi_{A,B} = P(X_{n+1} = B | X_n = A) \end{matrix}$$

FIGURE 5.20 – Exemple de Modèle de Markov à deux états

— $\vec{\nu}$, son vecteur d'initialisation exprimé en termes de probabilités de X de se trouver dans chaque état de E à l'instant initial.
— Π, sa matrice de transition exprimée en termes de probabilités pour X de passer à l'état suivant, quel qu'il soit, sachant l'état actuel, quel qu'il soit.
— B, sa matrice d'émission, exprimée en termes de probabilités de générer une observation, quelle qu'elle soit, sachant l'état X actuel, quel qu'il soit.

La distribution des observations s'exprime par $y_t = B \times vecx_t$.

Un exemple d'une Chaîne de Markov Cachée à deux états A et B et deux observations possibles 1 et 2 (discrètes dans notre exemple, mais peuvent être continues aussi) est présenté sur la Figure 5.21.

Inférence dans les Chaînes de Markov Cachées : l'algorithme *Forward-Backward*

L'objectif de cet algorithme est d'apprendre les estimées du vecteur d'initialisation $\widehat{\vec{\nu}}$, de la matrice de transition $\widehat{\Pi}$, et d'émission $\widehat{B}$ à partir de la suite des observations $y_{1:T}^{\vec{}}$, de l'espace d'état de l'état caché E, et de l'espace d'état des observations K.

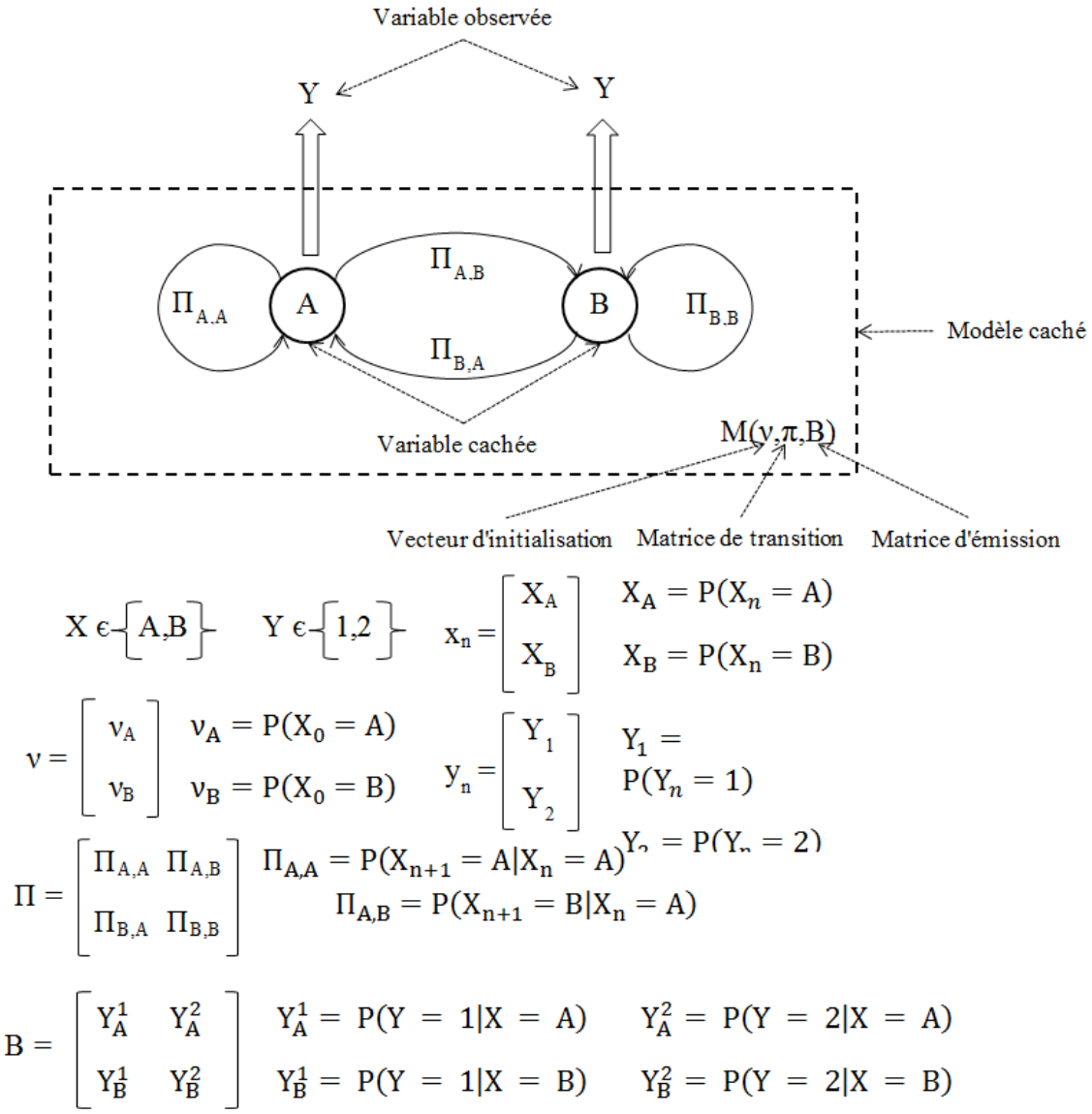


FIGURE 5.21 – Exemple de Modèle de Markov Caché à deux états et deux observations

Pour cela, on a besoin de six grandeurs :

— $\alpha_t(i), \forall t \in [1;T]$ et $\forall i \in E$
— $\beta_t(i), \forall t \in [1;T]$ et $\forall i \in E$
— $\xi_t(i,j), \forall t \in [1;T]$ et $\forall i,j \in E$
— $\gamma_t(i), \forall t \in [1;T]$ et $\forall i \in E$
— $x_t(i), \forall t \in [1;T]$ et $\forall i \in E$
— L

La signification de ces différentes grandeurs sera donnée lorsque nous présenterons

comment les calculer, un peu plus loin dans cette section.

Les probabilités précédentes sont pour la plupart rassemblées dans les vecteurs et matrices suivants pour être calculés plus simplement :

- $\vec{\alpha_t} = \{\alpha_t(i)\}_{i \in E}$
- $\vec{\beta_t} = \{\beta_t(i)\}_{i \in E}$
- $\xi_t = \{\xi_t(i,j)\}_{i,j \in E}$
- $\vec{\gamma_t} = \{\gamma_t(i)\}_{i \in E}$
- $\vec{x_t} = \{x_t(i)\}_{i \in E}$

Ces vecteurs et matrices sont ensuite rassemblés dans des matrices à deux ou trois dimensions pour simplifier leur manipulation :

- $\alpha = \{\vec{\alpha_t}\}_{t \in [1;T]}$
- $\beta = \{\vec{\beta_t}\}_{t \in [1;T]}$
- $\xi = \{\xi_t\}_{t \in [1;T]}$
- $\gamma = \{\vec{\gamma t}\}_{t \in [1;T]}$
- $x = \{\vec{x_t}\}_{t \in [1;T]}$

Le principe de l'algorithme est le suivant :

- Une phase d'initialisation visant à définir aléatoirement $\widehat{\vec{\nu}}$, $\widehat{\Pi}$ et $\widehat{B}$
- On itère ensuite jusqu'à diminution de la variation de L en-dessous du seuil ϵ :
  - Une phase *Forward* visant à calculer les $\vec{\alpha_t}$ à partir de $\widehat{\vec{\nu}}$, $\widehat{\Pi}$, $\hat{B}$ et de la suite des observations
  - Une phase *Backward* visant à calculer les $\vec{\beta_t}$ à partir de $\widehat{\Pi}$, $\widehat{B}$ et de la suite des observations
  - Une phase *Expectation* visant à calculer les ξ_t et $\vec{\gamma_t}$:
    - Calcul des ξ_t à partir des $\vec{\alpha_t}$, des $\vec{\beta_t}$, de $\widehat{\Pi}$, de $\widehat{B}$ et de la suite des observations
    - Calcul des $\vec{\gamma_t}$ à partir des ξ_t, ou des $\vec{\alpha_t}$ et $\vec{\beta_t}$
  - Une phase *Maximization* visant à redéfinir $\widehat{\vec{\nu}}$, $\widehat{\Pi}$ et $\widehat{B}$:
    - Calcul de $\widehat{\vec{\nu}}$ à partir de $\vec{\gamma_1}$
    - Calcul de $\widehat{\Pi}$ à partir des ξ_t et des $\vec{\gamma_t}$
    - Calcul de $\widehat{B}$ à partir des $\vec{\gamma_t}$
  - Une phase de calcul de L :
    - Calcul des $\vec{x_t}$ à partir de $\widehat{\vec{\nu}}$ et $\widehat{\Pi}$
    - Calcul de L à partir de $\vec{x_t}$ et $\widehat{B}$

On introduit $O_t(i) = P(y_t|X_t = i) = B(i, y_t)$, la probabilité d'émission de l'observation observée à l'instant t, sachant l'état à l'instant t. On rassemble ces probabilités sous forme d'une matrice diagonale $O_t = diag(O_t(i))$.

Calcul de $\vec{\alpha_t}$

On cherche à calculer les $\alpha_t(i) = P(X_t = i|y_{\vec{1:t}})$, la probabilité que l'état caché de la chaîne à l'instant t soit i, sachant la suite des observations jusqu'à l'instant t.

$$P(X_t = i|y_{\vec{1:t}}) = \frac{P(X_t = i, y_{\vec{1:t}})}{P(y_{\vec{1:t}})} \tag{5.2}$$

$$
\begin{aligned}
&= \frac{P(X_t = i, y_{1:t}^{\vec{}})}{\sum_{i \in E} P(X_t = i, y_{1:t}^{\vec{}})} \\
&= \frac{P(X_t = i, y_{1:\vec{t-1}}, y_t)}{\sum_{i \in E} P(X_t = i, y_{1:\vec{t-1}}, y_t)} \\
&= \frac{P(y_t | X_t = i, y_{1:\vec{t-1}}) P(X_t = i, y_{1:\vec{t-1}})}{\sum_{i \in E} P(y_t | X_t = i, y_{1:\vec{t-1}}) P(X_t = i, y_{1:\vec{t-1}})} \\
&= \frac{P(y_t | X_t = i) P(X_t = i, y_{1:\vec{t-1}})}{\sum_{i \in E} P(y_t | X_t = i) P(X_t = i, y_{1:\vec{t-1}})} \\
&= \frac{P(y_t | X_t = i) P(X_t = i | y_{1:\vec{t-1}}) P(y_{1:\vec{t-1}})}{\sum_{i \in E} P(y_t | X_t = i) P(X_t = i | y_{1:\vec{t-1}}) P(y_{1:\vec{t-1}})} \\
&= \frac{P(y_t | X_t = i) P(X_t = i | y_{1:\vec{t-1}})}{\sum_{i \in E} P(y_t | X_t = i) P(X_t = i | y_{1:\vec{t-1}})} \\
&= \frac{[\sum_{j \in E} P(X_t = i | X_{t-1} = j) P(X_{t-1} = j | y_{1:\vec{t-1}})] P(y_t | X_t = i)}{\sum_{i \in E} [\sum_{j \in E} P(X_t = i | X_{t-1} = j) P(X_{t-1} = j | y_{1:\vec{t-1}})] P(y_t | X_t = i)} \\
\alpha_t(i) &= \frac{[\sum_{j \in E} \Pi_{j,i} \alpha_{t-1}(j)] O_t(i)}{\sum_{i \in E} [\sum_{j \in E} \Pi_{j,i} \alpha_{t-1}(j)] O_t(i)} \\
\vec{\alpha_t} &= \frac{O_t \Pi^t \vec{\alpha_{t-1}}}{\sum_{i \in E} \alpha_t(i)}
\end{aligned}
$$

où ^{t} est l'opérateur transposition. À l'initialisation, $\vec{\alpha_1} = \frac{O_1 \vec{\nu}}{\sum_{i \in E} \alpha_1(i)}$

Calcul de $\vec{\beta_t}$

On cherche à calculer les $\beta_t(i) = P(y_{t+1:T} | X_t = i)$, la probabilité d'avoir la suite des observations de l'instant t+1 à T, sachant que l'état caché de la chaîne à l'instant t est i.

$$
\begin{aligned}
P(y_{t+1:T} | X_t = i) &= P(y_{t+1}, y_{t+2:T} | X_t = i) \\
&= \sum_{j \in E} P(y_{t+1}, y_{t+2:T}, X_{t+1} = j | X_t = i) \\
&= \sum_{j \in E} \frac{P(y_{t+1}, y_{t+2:T}, X_{t+1} = j, X_t = i)}{P(X_t = i)} \\
&= \sum_{j \in E} \frac{P(y_{t+2:T} | X_{t+1} = j, y_{t+1}, X_t = i) P(X_{t+1} = j, y_{t+1}, X_t = i)}{P(X_t = i)}
\end{aligned} \tag{5.3}
$$

$$
\begin{aligned}
&= \sum_{j\in E} \frac{P(y_{t+2:T}|X_{t+1}=j)P(X_{t+1}=j, y_{t+1}, X_t=i)}{P(X_t=i)} \\
&= \sum_{j\in E} \frac{P(y_{t+2:T}|X_{t+1}=j)P(y_{t+1}|X_{t+1}=j, X_t=i)P(X_{t+1}=j, X_t=i)}{P(X_t=i)} \\
&= \sum_{j\in E} \frac{P(y_{t+2:T}|X_{t+1}=j)P(y_{t+1}|X_{t+1}=j)P(X_{t+1}=j, X_t=i)}{P(X_t=i)} \\
&= \sum_{j\in E} \frac{P(y_{t+2:T}|X_{t+1}=j)P(y_{t+1}|X_{t+1}=j)P(X_{t+1}=j|X_t=i)P(X_t=i)}{P(X_t=i)} \\
&= \sum_{j\in E} P(y_{t+2:T}|X_{t+1}=j)P(y_{t+1}|X_{t+1}=j)P(X_{t+1}=j|X_t=i) \\
\beta_t(i) &= \sum_{j\in E} \beta_{t+1}(j)O_{t+1}(j)\Pi_{i,j} \\
\vec{\beta_t} &= \Pi O_{t+1}\vec{\beta_{t+1}}
\end{aligned}
$$

À l'initialisation, $\vec{\beta_T} = 1_{Card(E)}$, où $1_{Card(E)}$ représente un vecteur colonne de dimension Card(E) contenant exclusivement des 1.

Afin de prévenir le problème d'underflow, on normalise $\vec{\beta_t}$ à chaque itération.

Une phase *Backward* alternative

On utilise le fait que $X_{t-1} \perp y_{t+1:T}|X_t$, c'est à dire que sachant une valeur de l'état caché à un instant donné, la valeur de l'état caché à l'instant précédent est indépendante de la suite des observations des instants suivants.

$$
\begin{aligned}
P(X_{t-1}=i|y_{1:T}^{\vec{}}) &= \sum_{j\in E} P(X_{t-1}=i, X_t=j|y_{1:T}^{\vec{}}) \qquad (5.4)\\
&= \sum_{j\in E} \frac{P(X_{t-1}=i, X_t=j, y_{1:T}^{\vec{}})}{P(y_{1:T}^{\vec{}})} \\
&= \sum_{j\in E} \frac{P(X_{t-1}=i|X_t=j, y_{1:T}^{\vec{}})P(X_t=j, y_{1:T}^{\vec{}})}{P(y_{1:T}^{\vec{}})} \\
&= \sum_{j\in E} \frac{P(X_{t-1}=i|X_t=j, y_{1:T}^{\vec{}})P(X_t=j|y_{1:T}^{\vec{}})P(y_{1:T}^{\vec{}})}{P(y_{1:T}^{\vec{}})} \\
&= \sum_{j\in E} P(X_{t-1}=i|X_t=j, y_{1:T}^{\vec{}})P(X_t=j|y_{1:T}^{\vec{}}) \\
&= \sum_{j\in E} P(X_{t-1}=i|X_t=j, y_{1:t}^{\vec{}}, y_{t+1:T})P(X_t=j|y_{1:T}^{\vec{}}) \\
&= \sum_{j\in E} P(X_{t-1}=i|X_t=j, y_{1:t}^{\vec{}})P(X_t=j|y_{1:T}^{\vec{}}) \\
&= \sum_{j\in E} \frac{P(X_{t-1}=i, X_t=j, y_{1:t}^{\vec{}})P(X_t=j|y_{1:T}^{\vec{}})}{P(X_t=j, y_{1:t}^{\vec{}})}
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{j\in E} \frac{P(X_{t-1}=i, X_t=j, y_{1:t}^{\rightarrow}) P(X_t=j|y_{1:T}^{\rightarrow})}{P(X_t=j|y_{1:t}^{\rightarrow}) P(y_{1:t}^{\rightarrow})} \\
&= \sum_{j\in E} \frac{P(X_{t-1}=i, X_t=j, y_{1:t-1}^{\rightarrow}, y_t) P(X_t=j|y_{1:T}^{\rightarrow})}{P(X_t=j|y_{1:t}^{\rightarrow}) P(y_{1:t}^{\rightarrow})} \\
&= \sum_{j\in E} \frac{P(y_t|X_{t-1}=i, X_t=j, y_{1:t-1}^{\rightarrow}) P(X_{t-1}=i, X_t=j, y_{1:t-1}^{\rightarrow})}{P(X_t=j|y_{1:t}^{\rightarrow}) P(y_{1:t}^{\rightarrow})} \\
&\times P(X_t=j|y_{1:T}^{\rightarrow}) \\
&= \sum_{j\in E} \frac{P(y_t|X_t=j) P(X_{t-1}=i, X_t=j, y_{1:t-1}^{\rightarrow}) P(X_t=j|y_{1:T}^{\rightarrow})}{P(X_t=j|y_{1:t}^{\rightarrow}) P(y_{1:t}^{\rightarrow})} \\
&= \sum_{j\in E} \frac{P(y_t|X_t=j) P(X_t=j|X_{t-1}=i, y_{1:t-1}^{\rightarrow}) P(X_{t-1}=i, y_{1:t-1}^{\rightarrow})}{P(X_t=j|y_{1:t}^{\rightarrow}) P(y_{1:t}^{\rightarrow})} \\
&\times P(X_t=j|y_{1:T}^{\rightarrow}) \\
&= \sum_{j\in E} \frac{P(y_t|X_t=j) P(X_t=j|X_{t-1}=i) P(X_{t-1}=i, y_{1:t-1}^{\rightarrow}) P(X_t=j|y_{1:T}^{\rightarrow})}{P(X_t=j|y_{1:t}^{\rightarrow}) P(y_{1:t}^{\rightarrow})} \\
&= \sum_{j\in E} \frac{P(y_t|X_t=j) P(X_t=j|X_{t-1}=i) P(X_{t-1}=i|y_{1:t-1}^{\rightarrow}) P(y_{1:t-1}^{\rightarrow})}{P(X_t=j|y_{1:t}^{\rightarrow}) P(y_{1:t}^{\rightarrow})} \\
&\times P(X_t=j|y_{1:T}^{\rightarrow}) \\
\gamma_{t-1}(i) &= \sum_{j\in E} \frac{O_t(j) \Pi_{i,j} \alpha_{t-1}(i) P(y_{1:t-1}^{\rightarrow} y_{1:t-1}^{\rightarrow}) \gamma_t(j)}{\alpha_t(j) P(y_{1:t}^{\rightarrow})} \\
\gamma_{t-1}^{\rightarrow} &= diag(\alpha_{t-1}^{\rightarrow}) \Pi O_t \vec{\gamma_t} . * \vec{\alpha_t'}
\end{aligned}
$$

où $diag(\alpha_{t-1}^{\rightarrow})$ représente la matrice diagonale dont les éléments diagonaux sont les composantes du vecteur $\alpha_{t-1}^{\rightarrow}$, dans l'ordre, et $\vec{\alpha_t'} = \{\frac{1}{\alpha_t(i)}\}_{i\in E}$, le vecteur de l'inverse des composantes du vecteur $\vec{\alpha_t}$.

Calcul de ξ_t

On cherche à calculer les $\xi_t(i,j) = P(X_t=i, X_{t+1}=j|y_{1:T}^{\rightarrow})$, la probabilité que l'état caché vaille i à l'instant t et j à l'instant t+1, sachant la suite de l'intégralité des observations.

$$
\begin{aligned}
P(X_t=i, X_{t+1}=j|y_{1:T}^{\rightarrow}) &= \frac{P(X_t=i, X_{t+1}=j, y_{1:T}^{\rightarrow})}{P(y_{1:T}^{\rightarrow})} \\
&= \frac{P(X_t=i, X_{t+1}=j, y_{1:T}^{\rightarrow})}{\sum_{i\in E} P(X_t=i, y_{1:T}^{\rightarrow})} \\
&= \frac{P(X_t=i, X_{t+1}=j, y_{1:T}^{\rightarrow})}{\sum_{i\in E}\sum_{j\in E} P(X_t=i, X_{t+1}=j, y_{1:T}^{\rightarrow})} \\
&= \frac{P(X_t=i, X_{t+1}=j, y_{1:t}^{\rightarrow}, y_{t+1:T})}{\sum_{i\in E}\sum_{j\in E} P(X_t=i, X_{t+1}=j, y_{1:t}^{\rightarrow}, y_{t+1:T})}
\end{aligned}
\tag{5.5}
$$

$$
\begin{aligned}
&= \frac{P(y_{t+1:T}|X_t=i,X_{t+1}=j,y_{1:t}^{\rightarrow})P(X_t=i,X_{t+1}=j,y_{1:t}^{\rightarrow})}{\sum_{i\in E}\sum_{j\in E}P(y_{t+1:T}|X_t=i,X_{t+1}=j,y_{1:t}^{\rightarrow})P(X_t=i,X_{t+1}=j,y_{1:t}^{\rightarrow})}\\
&= \frac{P(y_{t+1:T}|X_{t+1}=j)P(X_t=i,X_{t+1}=j,y_{1:t}^{\rightarrow})}{\sum_{i\in E}\sum_{j\in E}P(y_{t+1:T}|X_{t+1}=j)P(X_t=i,X_{t+1}=j,y_{1:t}^{\rightarrow})}\\
&= \frac{P(y_{t+1:T}|X_{t+1}=j)P(X_{t+1}=j|X_t=i,y_{1:t}^{\rightarrow})P(X_t=i,y_{1:t}^{\rightarrow})}{\sum_{i\in E}\sum_{j\in E}P(y_{t+1:T}|X_{t+1}=j)P(X_{t+1}=j|X_t=i,y_{1:t}^{\rightarrow})P(X_t=i,y_{1:t}^{\rightarrow})}\\
&= \frac{P(y_{t+1:T}|X_{t+1}=j)P(X_{t+1}=j|X_t=i)P(X_t=i,y_{1:t}^{\rightarrow})}{\sum_{i\in E}\sum_{j\in E}P(y_{t+1:T}|X_{t+1}=j)P(X_{t+1}=j|X_t=i)P(X_t=i,y_{1:t}^{\rightarrow})}\\
&= \frac{P(y_{t+1},y_{t+2:T}|X_{t+1}=j)P(X_{t+1}=j|X_t=i)P(X_t=i,y_{1:t}^{\rightarrow})}{\sum_{i\in E}\sum_{j\in E}P(y_{t+1},y_{t+2:T}|X_{t+1}=j)P(X_{t+1}=j|X_t=i)P(X_t=i,y_{1:t}^{\rightarrow})}\\
&= \frac{P(y_{t+1}|X_{t+1}=j)P(y_{t+2:T}|X_{t+1}=j)P(X_{t+1}=j|X_t=i)}{\sum_{i\in E}\sum_{j\in E}P(y_{t+1}|X_{t+1}=j)P(y_{t+2:T}|X_{t+1}=j)P(X_{t+1}=j|X_t=i)}\\
&\times \frac{P(X_t=i,y_{1:t}^{\rightarrow})}{P(X_t=i,y_{1:t}^{\rightarrow})}\\
\xi_t(i,j) &= \frac{O_{t+1}(j)\beta_{t+1}(j)\Pi_{i,j}\alpha_t(i)}{\sum_{i\in E}\sum_{j\in E}O_{t+1}(j)\beta_{t+1}(j)\Pi_{i,j}\alpha_t(i)}\\
\xi_t &= \frac{diag(\vec{\alpha_t})\Pi O_{t+1}diag(\vec{\beta_{t+1}})}{\sum_{i,j\in E}\xi_t(i,j)}
\end{aligned}
$$

Calcul de $\vec{\gamma_t}$

On cherche à calculer les $\gamma_t(i) = P(X_t = i|y_{1:T}^{\rightarrow})$, la probabilité que l'état caché de la chaîne à l'instant t soit i, sachant la suite de l'intégralité des observations.

À partir de $\vec{\alpha_t}$ et $\vec{\beta_t}$

$$
\begin{aligned}
P(X_t=i|y_{1:T}^{\rightarrow}) &= \frac{P(X_t=i,y_{1:T}^{\rightarrow})}{P(y_{1:T}^{\rightarrow})} \qquad (5.6)\\
&= \frac{P(X_t=i,y_{1:T}^{\rightarrow})}{\sum_{i\in E}P(X_t=i,y_{1:T}^{\rightarrow})}\\
&= \frac{P(X_t=i,y_{1:t}^{\rightarrow},y_{t+1:T})}{\sum_{i\in E}P(X_t=i,y_{1:t}^{\rightarrow},y_{t+1:T})}\\
&= \frac{P(y_{t+1:T}|X_t=i,y_{1:t}^{\rightarrow})P(X_t=i,y_{1:t}^{\rightarrow})}{\sum_{i\in E}P(y_{t+1:T}|X_t=i,y_{1:t}^{\rightarrow})P(X_t=i,y_{1:t}^{\rightarrow})}\\
&= \frac{P(y_{t+1:T}|X_t=i)P(X_t=i,y_{1:t}^{\rightarrow})}{\sum_{i\in E}P(y_{t+1:T}|X_t=i)P(X_t=i,y_{1:t}^{\rightarrow})}
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{P(y_{t+1:T}|X_t = i)P(X_t = i|y_{1:t}^{\vec{}})P(y_{1:t}^{\vec{}})}{\sum_{i\in E} P(y_{t+1:T}|X_t = i)P(X_t = i|y_{1:t}^{\vec{}})P(y_{1:t}^{\vec{}})} \\
&= \frac{P(y_{t+1:T}|X_t = i)P(X_t = i|y_{1:t}^{\vec{}})}{\sum_{i\in E} P(y_{t+1:T}|X_t = i)P(X_t = i|y_{1:t}^{\vec{}})} \\
\gamma_t(i) &= \frac{\beta_t(i)\alpha_t(i)}{\sum_{i\in E} \beta_t(i)\alpha_t(i)} \\
\vec{\gamma_t} &= \frac{diag(\vec{\alpha_t})\vec{\beta_t}}{\sum_{i\in E} \gamma_t(i)}
\end{aligned}
$$

À partir de ξ_t

$$
\begin{aligned}
P(X_t = i|y_{1:T}^{\vec{}}) &= \sum_{j\in E} P(X_t = i, X_{t+1} = j|y_{1:T}^{\vec{}}) \\
\gamma_t(i) &= \sum_{j\in E} \xi_t(i,j)
\end{aligned}
\tag{5.7}
$$

Calcul de $\widehat{\vec{\nu}}$

On cherche à calculer les
$\widehat{\nu_i} = fréquence\ attendue\ de\ passage\ dans\ l'état\ i\ à\ l'instant\ 1\ sachant\ la\ suite\ des\ observations$,
l'estimée de la probabilité que l'état caché à l'instant 1 soit i.

$$
\begin{aligned}
\widehat{\nu(i)} &= P(X_1 = i|y_{1:T}^{\vec{}}) \\
&= \gamma_1(i) \\
\widehat{\vec{\nu}} &= \gamma_1
\end{aligned}
\tag{5.8}
$$

Calcul de $\widehat{\Pi}$

On cherche à calculer les
$\widehat{\Pi_{i,j}} = \frac{nombre\ attendu\ de\ transitions\ de\ l'état\ i\ vers\ l'état\ j\ sachant\ la\ suite\ des\ observations}{nombre\ attendu\ de\ transitions\ depuis\ l'état\ i\ sachant\ la\ suite\ des\ observations}$,
l'estimée de la probabilité que l'état caché de la chaîne à l'instant t+1 soit j, sachant que l'état caché de la chaîne à l'instant t était j.

$$
\widehat{\Pi_{i,j}} = \frac{\sum_{t=1}^{T-1} P(X_t = i, X_{t+1} = j|y_{1:T}^{\vec{}})}{\sum_{t=1}^{T-1} P(X_t = i|y_{1:T}^{\vec{}})}
\tag{5.9}
$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

Calcul de $\widehat{B}$

On cherche à calculer les
$\widehat{B_{i,k}} = \frac{\textit{nombre attendu d'observations de } y^k \textit{ dans l'état i sachant la suite des observations}}{\textit{nombre attendu de transitions depuis l'état i sachant la suite des observations}}$,
l'estimée de la probabilité que l'observation émise à l'instant t soit y^k ($k^{ème}$ observation de l'espace K des valeurs de Y), sachant que l'état caché de la chaîne à l'instant t est i.

$$\widehat{B_{i,k}} = \frac{\sum_{t=1}^{T} P(X_t = i|\vec{y_{1:T}}) 1(y_t = k)}{\sum_{t=1}^{T} P(X_t = i|\vec{y_{1:T}})} \tag{5.10}$$

$$= \frac{\sum_{t=1}^{T} \gamma_t(i) 1(y_t = k)}{\sum_{t=1}^{T} \gamma_t(i)}$$

$$\widehat{B} = diag((\gamma 1_T)')(\gamma(1(y_t = k))^t)$$

, où $1(y_t = k)$ représente une matrice de dimension $K \times T$ dont chaque élément $1_{k,t}$ est le résultat de la fonction indicatrice $y_t = k$, et l'opérateur ' inverse chaque élément de la matrice ou du vecteur auquel il est appliqué.

Calcul de $\vec{x_t}$

On cherche à calculer les $x_t(i) = P(X_t = i|\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})$, le vecteur de distribution de probabilité de l'état caché à l'instant t sachant le modèle estimé.

$$P(X_t = i|\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B}) = \sum_{j \in E} P(X_t = i, X_{t+1} = j|\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B}) \tag{5.11}$$

$$= \sum_{j \in E} \frac{P(X_t = i, X_{t+1} = j, \widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})}{P(\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})}$$

$$= \sum_{j \in E} \frac{P(X_t = i|X_{t+1} = j, \widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B}) P(X_{t+1} = j, \widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})}{P(\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})}$$

$$= \sum_{j \in E} \frac{P(X_t = i|X_{t+1} = j, \widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B}) P(X_{t+1} = j|\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B}) P(\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})}{P(\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})}$$

$$= \sum_{j \in E} P(X_t = i|X_{t+1} = j, \widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B}) P(X_{t+1} = j|\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})$$

$$x_t(i) = \sum_{j \in E} \widehat{\Pi_{j,i}} x_{t-1}(j)$$
$$\vec{x_t} = \widehat{\Pi}^t \vec{x_{t-1}}$$

À l'initialisation, $\vec{x_1} = \widehat{\vec{\nu}}$

Calcul de L

On cherche à calculer $L = P(\vec{y_{1:T}}|\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})$, la vraisemblance du modèle.

$$\begin{aligned}
P(\vec{y_{1:T}}|\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B}) &= \prod_1^T P(y_t|\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B}) \qquad (5.12)\\
&= \prod_1^T \sum_{i \in E} P(y_t, X_t = i|\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})\\
&= \prod_1^T \sum_{i \in E} \frac{P(y_t, X_t = i, \widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})}{P(\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})}\\
&= \prod_1^T \sum_{i \in E} \frac{P(y_t|X_t = i, \widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B}) P(X_t = i, \widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})}{P(\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})}\\
&= \prod_1^T \sum_{i \in E} \frac{P(y_t|X_t = i, \widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B}) P(X_t = i|\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B}) P(\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})}{P(\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})}\\
&= \prod_1^T \sum_{i \in E} P(y_t|X_t = i, \widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B}) P(X_t = i|\widehat{\vec{\nu}}, \widehat{\Pi}, \widehat{B})\\
&= \prod_1^T \sum_{i \in E} \widehat{O_t(i)} x_t(i)\\
L &= \prod_1^T 1^t_{Card(E)} \widehat{O_t} \vec{x_t}
\end{aligned}$$

Algorithme final

L'algorithme complet *Forward-Backward* est présenté dans l'Algorithme 12.

Algorithme 12 : Algorithme *Forward-Backward* final

Données : $\vec{y_{1:T}}$, E, K, ϵ
Résultat : $\widehat{\vec{\nu}}$, $\widehat{\Pi}$, $\widehat{B}$

Déterminer aléatoirement $\widehat{\vec{\nu}}$, $\widehat{\Pi}$ et $\widehat{B}$ (loi uniforme, par exemple);
tant que $\Delta L > \epsilon$ **faire**
  $\vec{\alpha_1} = \dfrac{O_1 \vec{\nu}}{\sum\limits_{i \in E} \alpha_1(i)}$;
  pour *t=2 à T* **faire**
    $\vec{\alpha_t} = \dfrac{O_t \Pi^t \vec{\alpha_{t-1}}}{\sum\limits_{i \in E} \alpha_t(i)}$;
  fin
  $\vec{\beta_T} = \{\dfrac{1}{Card(E)}\}_{Card(E)}$;
  $\vec{\gamma_T} = \dfrac{diag(\vec{\alpha_T})\vec{\beta_T}}{\sum\limits_{i \in E} \gamma_T(i)}$;
  $\xi' = 0_{Card(E),Card(E)}$;
  pour *t=T-1 à 1* **faire**
    $\vec{\beta_t} = \dfrac{\Pi O_{t+1} \vec{\beta_{t+1}}}{\sum\limits_{i \in E} \beta_t(i)}$;
    $\xi_t = \dfrac{diag(\vec{\alpha_t}) \Pi O_{t+1} diag(\vec{\beta_{t+1}})}{\sum\limits_{i,j \in E} \xi_t(i,j)}$;
    $\xi' = \xi' + \xi_t$;
    $\vec{\gamma_t} = \dfrac{diag(\vec{\alpha_t})\vec{\beta_t}}{\sum\limits_{i \in E} \gamma_t(i)}$;
  fin
  $\widehat{\vec{\nu}} = \gamma_1$;
  pour *i=1 à Card(E)* **faire**
    pour *j=1 à Card(E)* **faire**
      $\widehat{\Pi_{i,j}} = \dfrac{\xi(i,j)}{diag((\gamma_{1:T-1} 1_{T-1})')}$;
    fin
  fin
  $\widehat{B} = diag((\gamma 1_T)')(\gamma(1(y_t = k))^t)$;
  $\vec{x_1} = \widehat{\vec{\nu}}$;
  $L = 1^t_{Card(E)} \widehat{O_1} \vec{x_1}$;
  pour *t=2 à T* **faire**
    $\vec{x_t} = \widehat{\Pi}^t \vec{x_{t-1}}$;
    $L = L \times 1^t_{Card(E)} \widehat{O_t} \vec{x_t}$;
  fin
fin

Cet algorithme est illustré de façon graphique sur le schéma 5.22.

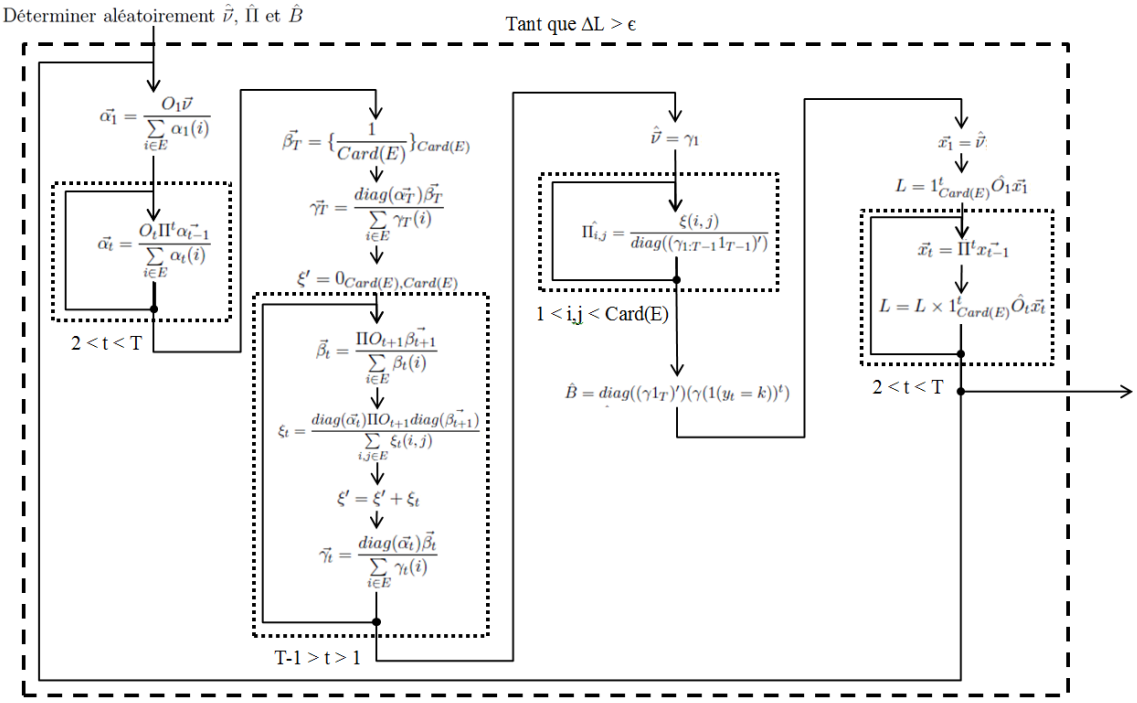


FIGURE 5.22 – Illustration graphique de l'algorithme *Forward-Backward*

Inférence dans les Chaînes de Markov Cachées : l'algorithme de Viterbi

L'objectif de cet algorithme est de déterminer $v^{\vec{max}}_{1:T}$, la suite d'états cachés la plus probable ayant généré la suite des observations de l'instant 1 à T, ainsi que P^{max}_T, la probabilité de cette suite d'observations, à partir du vecteur d'initialisation $\vec{\nu}$, de la matrice de transition Π, de la matrice d'émission B, et de la suite des observations $y_{\vec{1:T}}$.

Pour cela, on utilise les grandeurs $p_t(i), \forall t \in [1;T]$ et $\forall i \in E$. Chacune de ces grandeurs représente la probabilité de la suite d'états la plus probable aboutissant à l'état i à l'instant t. Elles sont rassemblées dans le vecteur $\vec{p_t} = \{p_t(i)\}_{i\in E}$. On utilise également les grandeurs $v_{1:t}(i), \forall t \in [1;T]$ et $\forall i \in E$. Chacune de ces grandeurs représente la suite d'états la plus probable aboutissant à l'état i à l'instant t. Elles sont rassemblées dans le vecteur $v_{\vec{1:t}} = \{v_{1:t}(i)\}_{i\in E}$.

$$\begin{aligned}(v^{\vec{max}}_{1:T}, P^{max}_T) &= \underset{X_{\vec{1:T}}}{\text{argmax}}, \underset{X_{\vec{1:T}}}{\max} P(X_{\vec{1:T}}, y_{\vec{1:T}}) \\ &= \underset{i}{\text{argmax}}, \underset{i}{\max}[\underset{X_{\vec{1:T-1}}}{\text{argmax}}, \underset{X_{\vec{1:T-1}}}{\max} P(X_{\vec{1:T-1}}, X_T = i, y_{\vec{1:T}})]\end{aligned} \tag{5.13}$$

$$= \underset{i}{\text{argmax}}, \underset{i}{\max}(v_{1:\vec{T}}(i), p_T(i))$$
$$= \underset{row}{\text{argmax}}, \underset{row}{\max}(v_{1:\vec{T}}, p_{\vec{T}})$$

$\underset{row}{\text{argmax}}, \underset{row}{\max}$ signifie que l'on cherche la valeur et l'indice de la ligne de plus forte valeur, pour chaque colonne de la matrice considérée.

Il est à noter que les opérations max et argmax ne sont appliquées qu'à $p_{\vec{T}}$, $v_{\vec{T}}$ n'étant présent que pour des facilités de suivi de la démonstration. Le même principe s'applique pour le reste des calculs.

$$\begin{aligned}
(v_{1:\vec{T}}(i), p_T(i)) &= \underset{X_{1:\vec{T}-1}}{\text{argmax}}, \underset{X_{1:\vec{T}-1}}{\max} P(X_{1:\vec{T}-1}, X_T = i, y_{1:\vec{T}}) \qquad (5.14)\\
&= \underset{X_{1:\vec{T}-1}}{\text{argmax}}, \underset{X_{1:\vec{T}-1}}{\max} P(X_{1:\vec{T}-1}, X_T = i, y_{1:\vec{T}-1}, y_T)\\
&= \underset{X_{1:\vec{T}-1}}{\text{argmax}}, \underset{X_{1:\vec{T}-1}}{\max} P(y_T|X_{1:\vec{T}-1}, X_T = i, y_{1:\vec{T}-1}) P(X_{1:\vec{T}-1}, X_T = i, y_{1:\vec{T}-1})\\
&= \underset{X_{1:\vec{T}-1}}{\text{argmax}}, \underset{X_{1:\vec{T}-1}}{\max} P(y_T|X_T = i) P(X_{1:\vec{T}-1}, X_T = i, y_{1:\vec{T}-1})\\
&= \underset{X_{1:\vec{T}-1}}{\text{argmax}}, \underset{X_{1:\vec{T}-1}}{\max} P(y_T|X_T = i) P(X_{1:\vec{T}-2}, X_{T-1} = j, X_T = i, y_{1:\vec{T}-1})\\
&= \underset{X_{1:\vec{T}-1}}{\text{argmax}}, \underset{X_{1:\vec{T}-1}}{\max} P(y_T|X_T = i) P(X_T = i|X_{1:\vec{T}-2}, X_{T-1} = j, y_{1:\vec{T}-1})\\
&\times P(X_{1:\vec{T}-2}, X_{T-1} = j, y_{1:\vec{T}-1})\\
&= \underset{X_{1:\vec{T}-1}}{\text{argmax}}, \underset{X_{1:\vec{T}-1}}{\max} P(y_T|X_T = i) P(X_T = i|X_{T-1} = j)\\
&\times P(X_{1:\vec{T}-2}, X_{T-1} = j, y_{1:\vec{T}-1})\\
&= \underset{j}{\text{argmax}}, \underset{j}{\max}[\underset{X_{1:\vec{T}-2}}{\text{argmax}}, \underset{X_{1:\vec{T}-2}}{\max} P(y_T|X_T = i) P(X_T = i|X_{T-1} = j)\\
&\times P(X_{1:\vec{T}-2}, X_{T-1} = j, y_{1:\vec{T}-1})]\\
&= \underset{j}{\text{argmax}}, \underset{j}{\max}[P(y_T|X_T = i) P(X_T = i|X_{T-1} = j) \underset{X_{1:\vec{T}-2}}{\text{argmax}}, \underset{X_{1:\vec{T}-2}}{\max}\\
&\times P(X_{1:\vec{T}-2}, X_{T-1} = j, y_{1:\vec{T}-1})]\\
&= \underset{j}{\text{argmax}}, \underset{j}{\max} O_T(i) \Pi_{j,i} (v_{1:\vec{T-1}}(j), p_{T-1}(j))\\
(v_{1:\vec{T}}, p_{\vec{T}}) &= \underset{column}{\text{argmax}}, \underset{column}{\max} O_T \Pi^t (v_{1:\vec{T}-1}, diag(p_{\vec{T-1}}))\\
&\textit{On en déduit que } \forall t \in [1; T]\,,\\
(v_{1:\vec{t}}, p_{\vec{t}}) &= \underset{column}{\text{argmax}}, \underset{column}{\max} O_t \Pi^t (v_{1:\vec{t}-1}, diag(p_{\vec{t-1}}))
\end{aligned}$$

À l'initialisation, $p_{\vec{1}} = O_1 \vec{\nu}$ et $v_{\vec{1}} = \emptyset$.

$\underset{column}{\text{argmax}}$, $\underset{column}{\max}$ signifie que l'on cherche la valeur et l'indice de la colonne de plus forte valeur, pour chaque ligne de la matrice considérée.

Afin, de prévenir le problème d'underflow, il est conseillé d'appliquer le logarithme népérien à chaque itération.

Algorithme final

L'algorithme de Viterbi complet est présenté dans l'Algorithme 13.

Algorithme 13 : Algorithme de Viterbi final

Données : $\vec{\nu}$, Π, B, $y_{1:T}$
Résultat : $v_{1:T}^{max}$,P_T^{max}

1 On initialise $\vec{p_1} = O_1\vec{\nu}$;
2 On initialise $\vec{v_1} = \emptyset$;
3 **pour** *t=2 à T* **faire**
4 $\quad (\vec{v_{1:t}}, \vec{p_t}) = \underset{column}{\text{argmax}}, \underset{column}{\max}\ O_t\Pi^t(v_{1:\vec{t-1}}, diag(\vec{p_{t-1}}))$;
5 **fin**
6 $(v_{1:T}^{max}, P_T) = \underset{row}{\text{argmax}}, \underset{row}{\max}(\vec{v_{1:T}}, \vec{p_T})$;

Cet algorithme est illustré de façon graphique sur le schéma 5.23.

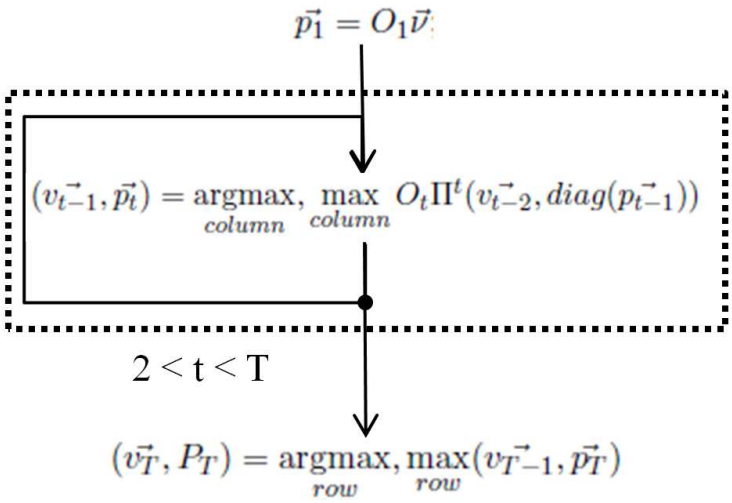


FIGURE 5.23 – Illustration graphique de l'algorithme de Viterbi

Annexe 3 : Échantillonneur de Gibbs

L'échantillonneur de Gibbs est une technique issue des Chaînes de Markov Monte-Carlo (MCMC), elle même issue des simulations de Monte-Carlo. Nous présenterons donc d'abord brièvement ces deux techniques.

Simulations de Monte-Carlo

Il s'agit d'une famille de techniques visant à estimer numériquement une grandeur à l'aide de procédés aléatoires. Par exemple, si l'on considère une surface A scindée en deux parties A_1 et A_2, et que l'on cherche à déterminer le rapport $R = \frac{A_1}{A_2}$, alors on peut tirer aléatoirement des points selon une loi uniforme bidimensionnelle (Une composante pour chaque dimension de A) Si l'on note N_1 le nombre de points tombés dans A_1 et N_2 le nombre de points tombés dans A_2, alors $R \approx \frac{N_1}{N_2}$ pour un nombre N de tirage suffisamment grand, et $\lim_{N \to +\infty} \frac{N_1}{N_2} = R$.

Sur l'exemple suivant, N = 12, $N_1 = 9$, $N_2 = 3$, donc $R \approx 3$, ce qui ne parait pas trop loin de la réalité (2.82).

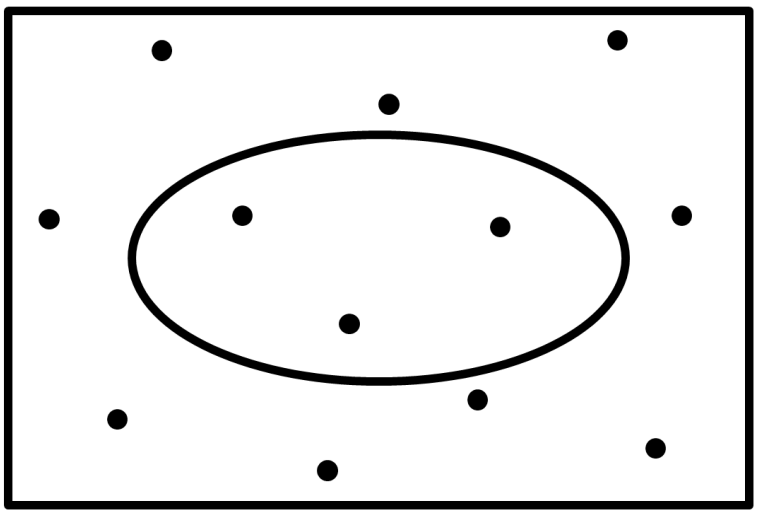


FIGURE 5.24 – Exemple d'approximation de rapport de surfaces via simulation de Monte-Carlo

Chaînes de Markov de Monte-Carlo (MCMC)

Il s'agit d'une famille de techniques visant à estimer les propriétés statistiques d'une distribution de probabilités. Pour cela, des valeurs aléatoires suivant la distribution recherchée sont simulées à l'aide d'une Chaîne de Markov.

Cette dernière doit être fortement connexe, c'est à dire qu'il existe un chemin vers n'importe quel état depuis un état donné, et ce pour tout état. Cette propriété permet d'assurer que la distribution de probabilité des états de la chaîne à un instant donné converge vers une distribution unique fixe Π' dépendant de la chaîne. De plus, si l'on considère la moyenne du vecteur d'états x sur un grand nombre d'itérations, alors cette distribution correspond à Π'.

D'un point de vue mathématique, en reprenant les notations précédentes, on a :
— $\Pi' \times \Pi = \Pi'$
— $\bar{x_n} = \frac{1}{n} \sum_{i=1}^{n} x_i,\ et \lim_{n \to +\infty} \bar{x_n} = \Pi'$

Échantillonneur de Gibbs

L'échantillonneur de Gibbs est un algorithme issu des MCMC visant à simuler un échantillonnage d'une distribution a posteriori trop complexe pour être manipulée formellement. Il est ainsi possible d'extraire les grandeurs statistiques de la distribution à partir des valeurs générées, sans avoir besoin de les calculer de façon exacte à partir de la formule de leur densité de probabilités.

Nous allons présenter le mode opératoire de création d'un échantillonneur de Gibbs.

On pose :
— X, réalisation de la loi (potentiellement multi-dimensionnel)
— $\vec{\theta}$, jeu de paramètres de la loi
— $\vec{\theta} = \{\theta_j\}_{1 \leq j \leq p}$
— $\vec{\theta_{-i}} = \{\theta_j\}_{1 \leq j \leq p, j \neq i}$

Opérations préliminaires :

1. Définir le modèle :
   — Vraisemblance : $P(X|\vec{\theta})$
   — Lois a priori : $P(\theta_i)$
2. Calculer la distribution conjointe : $P(X, \vec{\theta})$
3. Calculer les distributions conditionnelles : $P(\theta_i|\vec{\theta_{-i}}, X)$

Implémentation de l'échantillonneur (voir schéma 5.25) :

1. Initialiser X et θ à une valeur quelconque de l'univers des possibles : $(X^0, \vec{\theta^0})$
2. Répéter jusqu'à convergence :
   (a) Échantillonner $X^n \sim P(X|\vec{\theta^{n-1}})$

(b) Échantillonner $\theta_i^n \sim P(\theta_i|\vec{\theta_{-i}^{n-1}}, X^n)$

Une fois la convergence atteinte (critère laissé à la discrétion de l'utilisateur), toutes les valeurs subséquemment générées respectent les lois a posteriori cibles $P(\theta_i|X)$. Avec un nombre d'échantillons suffisamment grand, il est possible d'approximer numériquement n'importe quelle statistique de la distribution.

Il est à noter que généralement, la vraisemblance et les lois a priori sont choisies de façon à être conjuguées, c'est à dire que la distribution a posteriori est issue de la même famille de distribution que la loi a priori, seuls les paramètres changent. cela permet de simplifier les calculs.

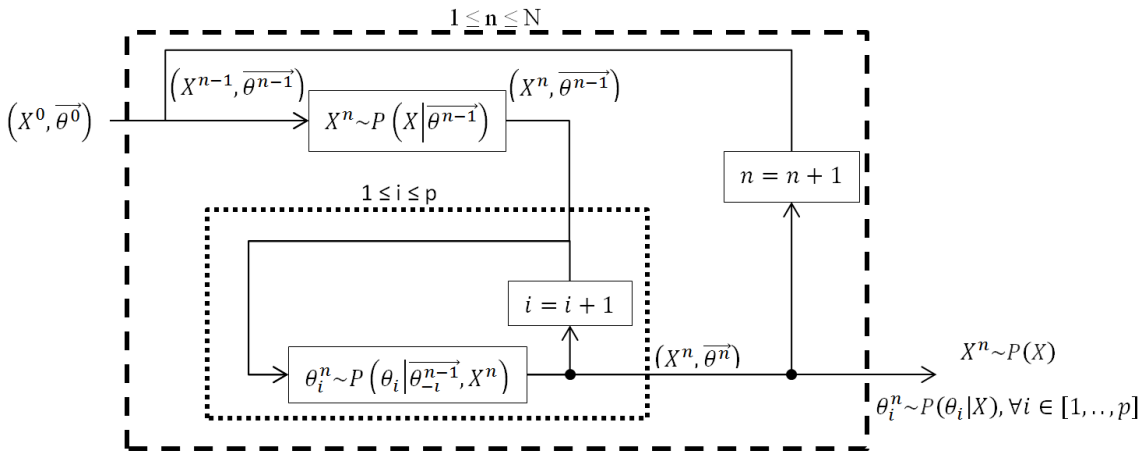


FIGURE 5.25 – Schéma de fonctionnement de l'échantillonneur de Gibbs

X^0 et θ^0 sont généralement initialisés par un tirage d'une loi uniforme sur leurs univers respectifs. N représente le nombre d'itérations demandées, supérieur au nombre nécessaire à la convergence.

De cette modélisation de l'échantillonneur de Gibbs, on peut en extraire une plus générique (voir schéma 5.26) en posant $\vec{\phi} = \left\{\vec{\theta}, X\right\}$, et en considérant que la variable à mettre à jour à chaque itération peut être déterminée de la façon souhaitée.

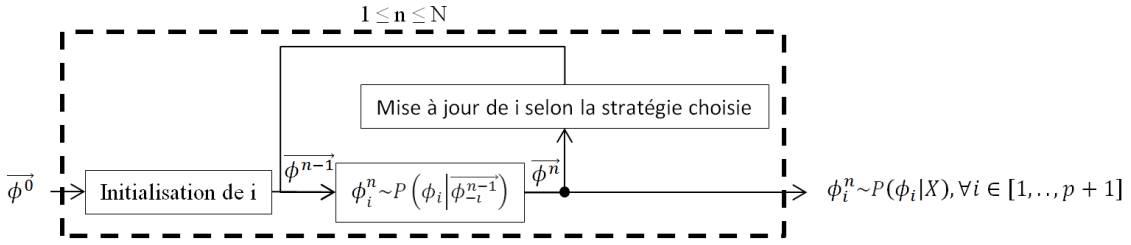


FIGURE 5.26 – Schéma de fonctionnement de l'échantillonneur de Gibbs générique

Annexe 4 : Inférence exacte dans les réseaux Bayésiens avec l'algorithme *Message Passing*

Nous allons appliquer l'algorithme de *Message Passing* à l'exemple présenté section 3.1.2. Les éléments introduits à cet endroit seront repris dans cette annexe, il est donc indispensable de les avoir vus au préalable. Pour cela, nous décidons que toutes les variables sont binaires, et que les nœuds A, D, et G sont observés, tandis que les nœuds B, C, et F sont cachés, comme illustré sur le schéma 5.27. Pour simplifier les notations, $P(\bar{X})$ signifie $P(X = 0)$, et $P(X)$ signifie $P(X = 1)$.

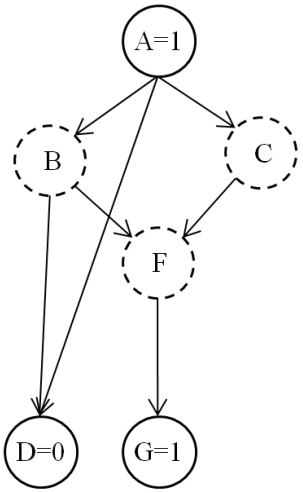


FIGURE 5.27 – Schéma du réseau bayésien utilisé pour l'exemple de *Message Passing*

Nous définissons les probabilités a priori et conditionnelles suivantes :

i	$P(A = i)$
0	$P(\bar{A})$
1	$P(A)$

$P(B = i\|A = j)$		
j \ i	0	1
0	$P(\bar{B}\|\bar{A})$	$P(B\|\bar{A})$
1	$P(\bar{B}\|A)$	$P(B\|A)$

$P(C=i\|A=j)$		
j \ i	0	1
0	$P(\bar{C}\|\bar{A})$	$P(C\|\bar{A})$
1	$P(\bar{C}\|A)$	$P(C\|A)$

$P(D=i\|A=j,B=k)$		
j,k \ i	0	1
00	$P(\bar{D}\|\bar{A},\bar{B})$	$P(D\|\bar{A},\bar{B})$
01	$P(\bar{D}\|\bar{A},B)$	$P(D\|\bar{A},B)$
10	$P(\bar{D}\|A,\bar{B})$	$P(D\|A,\bar{B})$
11	$P(\bar{D}\|A,B)$	$P(D\|A,B)$

$P(F=i\|B=j,C=k)$		
j,k \ i	0	1
00	$P(\bar{F}\|\bar{B},\bar{C})$	$P(F\|\bar{B},\bar{C})$
01	$P(\bar{F}\|\bar{B},C)$	$P(F\|\bar{B},C)$
10	$P(\bar{F}\|B,\bar{C})$	$P(F\|B,\bar{C})$
11	$P(\bar{F}\|B,C)$	$P(F\|B,C)$

$P(G=i\|F=j)$		
j \ i	0	1
0	$P(\bar{G}\|\bar{F})$	$P(G\|\bar{F})$
1	$P(\bar{G}\|F)$	$P(G\|F)$

Le schéma 5.28 illustre l'affectation des probabilités aux différentes cliques du *junction tree*.

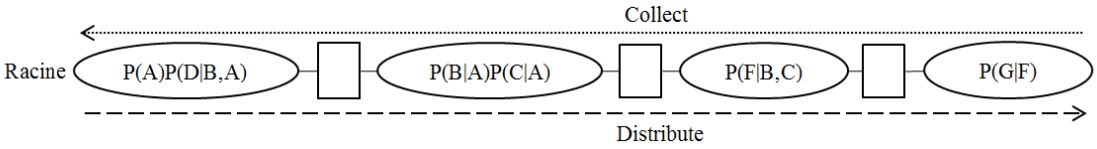


FIGURE 5.28 – Affectation des potentiels initiaux des cliques et des séparateurs

La variable A est observée à 1, la variable D à 0, et la variable G à 1, par conséquent, la probabilité que ces variables ne valent pas la valeur observée est de 0. Les autres valeurs sont laissées telles quelles. On initialise donc les potentiels des cliques comme suit :

$\psi^0_{ABD} = P(A=i)P(D=j\|A=i,B=k)$		
j,k \ i	0	1
00	0	$P(A)P(\bar{D}\|A,\bar{B})$
01	0	$P(A)P(\bar{D}\|A,B)$
10	0	0
11	0	0

$\psi^0_{ABC} = P(B=i\|A=j)P(C=k\|A=j)$		
j,k \ i	0	1
00	0	0
01	0	0
10	$P(\bar{B}\|A)P(\bar{C}\|A)$	$P(B\|A)P(\bar{C}\|A)$
11	$P(\bar{B}\|A)P(C\|A)$	$P(B\|A)P(C\|A)$

$\psi^0_{BCF} = P(F=i\|B=j,C=k)$		
j,k \ i	0	1
00	$P(\bar{F}\|\bar{B},\bar{C})$	$P(F\|\bar{B},\bar{C})$
01	$P(\bar{F}\|\bar{B},C)$	$P(F\|\bar{B},C)$
10	$P(\bar{F}\|B,\bar{C})$	$P(F\|B,\bar{C})$
11	$P(\bar{F}\|B,C)$	$P(F\|B,C)$

$\psi^0_{FG} = P(G=i\|F=j)$		
j \ i	0	1
0	0	$P(G\|\bar{F})$
1	0	$P(G\|F)$

Nous allons maintenant dérouler l'algorithme de *Message Passing* :
Passe *Collect* :

$\psi^c_{FG} = 1 \times \psi^0_{FG}$		
F \ G	0	1
0	0	$P(G\|\bar{F})$
1	0	$P(G\|F)$

F	$\mu_{FG\to BCF} = \sum_{FG\setminus F} \psi^c_{FG} = \sum_G \psi^c_{FG}$
0	$P(G\|\bar{F})$
1	$P(G\|F)$

$\psi^c_{BCF} = \mu_{FG\to BCF}\psi^0_{BCF}$		
B,C \ F	0	1
00	$P(G\|\bar{F})P(\bar{F}\|\bar{B},\bar{C})$	$P(G\|F)P(F\|\bar{B},\bar{C})$
01	$P(G\|\bar{F})P(\bar{F}\|\bar{B},C)$	$P(G\|F)P(F\|\bar{B},C)$
10	$P(G\|\bar{F})P(\bar{F}\|B,\bar{C})$	$P(G\|F)P(F\|B,\bar{C})$
11	$P(G\|\bar{F})P(\bar{F}\|B,C)$	$P(G\|F)P(F\|B,C)$

$\mu_{BCF\to ABC} = \sum_{BCF\setminus BC} \psi^c_{BCF} = \sum_F \psi^c_{BCF}$		
C \ B	0	1
0	$P(G\|\bar{F})P(\bar{F}\|\bar{B},\bar{C}) + P(G\|F)P(F\|\bar{B},\bar{C})$	$P(G\|\bar{F})P(\bar{F}\|B,\bar{C}) + P(G\|F)P(F\|B,\bar{C})$
1	$P(G\|\bar{F})P(\bar{F}\|\bar{B},C) + P(G\|F)P(F\|\bar{B},C)$	$P(G\|\bar{F})P(\bar{F}\|B,C) + P(G\|F)P(F\|B,C)$

B,A,C	$\psi^c_{ABC} = \mu_{BCF\to ABC}\psi^0_{ABC}$
000	0
001	0
010	$(P(G\|\bar{F})P(\bar{F}\|\bar{B},\bar{C}) + P(G\|F)P(F\|\bar{B},\bar{C}))P(\bar{B}\|A)P(\bar{C}\|A)$
011	$(P(G\|\bar{F})P(\bar{F}\|\bar{B},C) + P(G\|F)P(F\|\bar{B},C))P(\bar{B}\|A)P(C\|A)$
100	0
101	0
110	$(P(G\|\bar{F})P(\bar{F}\|B,\bar{C}) + P(G\|F)P(F\|B,\bar{C}))P(B\|A)P(\bar{C}\|A)$
111	$(P(G\|\bar{F})P(\bar{F}\|B,C) + P(G\|F)P(F\|B,C))P(B\|A)P(C\|A)$

B,A	$\mu_{ABC\to ABD} = \sum_{ABC\setminus AB} \psi^c_{ABC} = \sum_C \psi^c_{ABC}$
00	0
01	$(P(G\|\bar{F})P(\bar{F}\|\bar{B},\bar{C}) + P(G\|F)P(F\|\bar{B},\bar{C}))P(\bar{B}\|A)P(\bar{C}\|A)$ + $(P(G\|\bar{F})P(\bar{F}\|\bar{B},C) + P(G\|F)P(F\|\bar{B},C))P(\bar{B}\|A)P(C\|A)$
10	0
11	$(P(G\|\bar{F})P(\bar{F}\|B,\bar{C}) + P(G\|F)P(F\|B,\bar{C}))P(B\|A)P(\bar{C}\|A)$ + $(P(G\|\bar{F})P(\bar{F}\|B,C) + P(G\|F)P(F\|B,C))P(B\|A)P(C\|A)$

A, D, B	$\psi^c_{ABD} = \mu_{ABC \to ABD} \psi^0_{ABD}$
000	0
001	0
010	0
011	0
100	$(P(G\|\bar{F})P(\bar{F}\|\bar{B},\bar{C}) + P(G\|F)P(F\|\bar{B},\bar{C}))P(\bar{B}\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,\bar{B})$ $+$ $(P(G\|\bar{F})P(\bar{F}\|\bar{B},C) + P(G\|F)P(F\|\bar{B},C))P(\bar{B}\|A)P(C\|A)P(A)P(\bar{D}\|A,\bar{B})$
101	$(P(G\|\bar{F})P(\bar{F}\|B,\bar{C}) + P(G\|F)P(F\|B,\bar{C}))P(B\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,B)$ $+$ $(P(G\|\bar{F})P(\bar{F}\|B,C) + P(G\|F)P(F\|B,C))P(B\|A)P(C\|A)P(A)P(\bar{D}\|A,B)$
110	0
111	0

Passe *Distribute* (on s'abstiendra de normaliser pour des raisons de lisibilité) :

A, D, B	$\psi^d_{ABD} = 1 \times \psi^c_{ABD}$
000	0
001	0
010	0
011	0
100	$(P(G\|\bar{F})P(\bar{F}\|\bar{B},\bar{C}) + P(G\|F)P(F\|\bar{B},\bar{C}))P(\bar{B}\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,\bar{B})$ $+$ $(P(G\|\bar{F})P(\bar{F}\|\bar{B},C) + P(G\|F)P(F\|\bar{B},C))P(\bar{B}\|A)P(C\|A)P(A)P(\bar{D}\|A,\bar{B})$
101	$(P(G\|\bar{F})P(\bar{F}\|B,\bar{C}) + P(G\|F)P(F\|B,\bar{C}))P(B\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,B)$ $+$ $(P(G\|\bar{F})P(\bar{F}\|B,C) + P(G\|F)P(F\|B,C))P(B\|A)P(C\|A)P(A)P(\bar{D}\|A,B)$
110	0
111	0

$\mu_{ABD \to ABC} = \sum_{ABD \setminus AB} \frac{\psi^d_{ABD}}{\mu_{ABC \to ABD}} = \sum_D \frac{\psi^d_{ABD}}{\mu_{ABC \to ABD}}$

B \ A	0	1
0	0	$P(A)P(\bar{D}\|A,\bar{B})$
1	0	$P(A)P(\bar{D}\|A,B)$

B, A, C	$\psi^d_{ABC} = \mu_{ABD \to ABC} \psi^c_{ABC}$
000	0
001	0
010	$(P(G\|\bar{F})P(\bar{F}\|\bar{B},\bar{C}) + P(G\|F)P(F\|\bar{B},\bar{C}))P(\bar{B}\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,\bar{B})$
011	$(P(G\|\bar{F})P(\bar{F}\|\bar{B},C) + P(G\|F)P(F\|\bar{B},C))P(\bar{B}\|A)P(C\|A)P(A)P(\bar{D}\|A,\bar{B})$
100	0
101	0
110	$(P(G\|\bar{F})P(\bar{F}\|B,\bar{C}) + P(G\|F)P(F\|B,\bar{C}))P(B\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,B)$
111	$(P(G\|\bar{F})P(\bar{F}\|B,C) + P(G\|F)P(F\|B,C))P(B\|A)P(C\|A)P(A)P(\bar{D}\|A,B)$

$\mu_{ABC \to BCF} = \sum_{ABC \setminus BC} \frac{\psi^d_{ABC}}{\mu_{BCF \to ABC}} = \sum_{A} \frac{\psi^d_{ABC}}{\mu_{BCF \to ABC}}$		
C \ B	0	1
0	$P(\bar{B}\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,\bar{B})$	$P(B\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,B)$
1	$P(\bar{B}\|A)P(C\|A)P(A)P(\bar{D}\|A,\bar{B})$	$P(B\|A)P(C\|A)P(A)P(\bar{D}\|A,B)$

$\psi^d_{BCF} = \mu_{ABC \to BCF} \psi^c_{BCF}$		
B, C \ F	0	1
00	$P(G\|\bar{F})P(\bar{F}\|\bar{B},\bar{C})$ $\times$ $P(\bar{B}\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,\bar{B})$	$P(G\|F)P(F\|\bar{B},\bar{C})$ $\times$ $P(\bar{B}\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,\bar{B})$
01	$P(G\|\bar{F})P(\bar{F}\|\bar{B},C)$ $\times$ $P(\bar{B}\|A)P(C\|A)P(A)P(\bar{D}\|A,\bar{B})$	$P(G\|F)P(F\|\bar{B},C)$ $\times$ $P(\bar{B}\|A)P(C\|A)P(A)P(\bar{D}\|A,\bar{B})$
10	$P(G\|\bar{F})P(\bar{F}\|B,\bar{C})$ $\times$ $P(B\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,B)$	$P(G\|F)P(F\|B,\bar{C})$ $\times$ $P(B\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,B)$
11	$P(G\|\bar{F})P(\bar{F}\|B,C)$ $\times$ $P(B\|A)P(C\|A)P(A)P(\bar{D}\|A,B)$	$P(G\|F)P(F\|B,C)$ $\times$ $P(B\|A)P(C\|A)P(A)P(\bar{D}\|A,B)$

F	$\mu_{BCF\to FG} = \sum_{BCF\setminus F} \frac{\psi^d_{BCF}}{\mu_{FG\to BCF}} = \sum_{BC} \frac{\psi^d_{BCF}}{\mu_{FG\to BCF}}$
0	$P(\bar{F}\|\bar{B},\bar{C})P(\bar{B}\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,\bar{B})$ $+$ $P(\bar{F}\|\bar{B},C)P(\bar{B}\|A)P(C\|A)P(A)P(\bar{D}\|A,\bar{B})$ $+$ $P(\bar{F}\|B,\bar{C})P(B\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,B)$ $+$ $P(\bar{F}\|B,C)P(B\|A)P(C\|A)P(A)P(\bar{D}\|A,B)$
1	$P(F\|\bar{B},\bar{C})P(\bar{B}\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,\bar{B})$ $+$ $P(F\|\bar{B},C)P(\bar{B}\|A)P(C\|A)P(A)P(\bar{D}\|A,\bar{B})$ $+$ $P(F\|B,\bar{C})P(B\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,B)$ $+$ $P(F\|B,C)P(B\|A)P(C\|A)P(A)P(\bar{D}\|A,B)$

$\psi^d_{FG} = \mu_{BCF\to FG}\psi^c_{FG}$		
F \ G	0	1
0	0	$P(G\|\bar{F})P(\bar{F}\|\bar{B},\bar{C})P(\bar{B}\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,\bar{B})$ $+$ $P(G\|\bar{F})P(\bar{F}\|\bar{B},C)P(\bar{B}\|A)P(C\|A)P(A)P(\bar{D}\|A,\bar{B})$ $+$ $P(G\|\bar{F})P(\bar{F}\|B,\bar{C})P(B\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,B)$ $+$ $P(G\|\bar{F})P(\bar{F}\|B,C)P(B\|A)P(C\|A)P(A)P(\bar{D}\|A,B)$
1	0	$P(G\|F)P(F\|\bar{B},\bar{C})P(\bar{B}\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,\bar{B})$ $+$ $P(G\|F)P(F\|\bar{B},C)P(\bar{B}\|A)P(C\|A)P(A)P(\bar{D}\|A,\bar{B})$ $+$ $P(G\|F)P(F\|B,\bar{C})P(B\|A)P(\bar{C}\|A)P(A)P(\bar{D}\|A,B)$ $+$ $P(G\|F)P(F\|B,C)P(B\|A)P(C\|A)P(A)P(\bar{D}\|A,B)$

Annexe 5 : Détails sur l'obtention des formules de l'EM appliqué aux réseaux Bayésiens

Nous représenterons un paramètre du réseau sachant ses parents par $\theta_{i,j_i,k_i} = P(X_i = k_i|par(X_i) = j_i)$, avec i la variable considérée, j_i la valeur de ses parents considérée, et k_i la valeur de la variable considérée.

L'ensemble des nœuds du réseau bayésien comportant N variables est représenté par $X = (\, X_1 \;\; ... \;\; X_N \,)$

La probabilité conjointe (ou vraisemblance) de tous les nœuds de ce réseau est :

$$\begin{aligned} P(X;\Theta) &= \prod_{i=1}^{N} P(X_i|par(X_i)) \\ &= \prod_{i=1}^{N} \theta_{i,par(X_i),X_i} \end{aligned} \tag{5.15}$$

où Θ représente l'ensemble des paramètres du réseau. Pour éviter l'underflow lors de l'implémentation pratique de l'algorithme, on utilisera la log-vraisemblance, définie comme suit :

$$\begin{aligned} L(X;\Theta) = log(P(X;\Theta)) &= log(\prod_{i=1}^{N} \theta_{i,par(X_i),X_i}) \\ &= \sum_{i=1}^{N} log(\theta_{i,par(X_i),X_i}) \end{aligned} \tag{5.16}$$

On considère maintenant que l'on dispose d'un ensemble de M réalisations indépendantes de notre réseau bayésien. L'ensemble de ces réalisations est représenté par :

$$D = \begin{pmatrix} X_1^1 & ... & X_N^1 \\ ... & ... & ... \\ X_1^M & ... & X_N^M \end{pmatrix} \tag{5.17}$$

Grâce à l'indépendance des réalisations, la probabilité conjointe (ou vraisemblance) se décompose en un produit :

$$P(D;\Theta) = \prod_{m=1}^{M} P(X^m;\Theta) = \prod_{i=1}^{N} \theta_{i,par(X_i^m),X_i^m} \tag{5.18}$$

On obtient ensuite la log-vraisemblance suivante :

$$
\begin{aligned}
L(D;\Theta) &= log(P(D;\Theta)) \\
&= log(\prod_{m=1}^{M}\prod_{i=1}^{N}\theta_{i,par(X_i^m),X_i^m}) \\
&= \sum_{m=1}^{M}\sum_{i=1}^{N} log(\theta_{i,par(X_i^m),X_i^m}) \\
&= \sum_{m=1}^{M}\sum_{i=1}^{N} log(\prod_{j_i,k_i}\theta_{i,j_i,k_i}^{\mathbb{1}_{par(X_i^m)=j_i,X_i^m=k_i}}) \\
&= \sum_{m=1}^{M}\sum_{i=1}^{N}\sum_{j_i,k_i}\mathbb{1}_{par(X_i^m)=j_i,X_i^m=k_i} log(\theta_{i,j_i,k_i}) \\
&= \sum_{i,j_i,k_i} log(\theta_{i,j_i,k_i})\sum_{m=1}^{M}\mathbb{1}_{par(X_i^m)=j_i,X_i^m=k_i} \\
&= \sum_{i,j_i,k_i} log(\theta_{i,j_i,k_i}) N_{i,j_i,k_i}
\end{aligned}
\tag{5.19}
$$

N_{i,j_i,k_i} représente le nombre de fois où la variable X_i vaut k_i et ses parents valent j_i. Une partie des nœuds n'étant pas observée, N_{i,j_i,k_i} n'est pas calculable directement.

La fonction Q de l'algorithme EM s'écrit donc :

$$
\begin{aligned}
Q(\Theta,\Theta^r) &= \mathbb{E}(L(X;\Theta)|^oX,\Theta^r) \\
&= \mathbb{E}(\sum_{i,j_i,k_i} log(\theta_{i,j_i,k_i})\sum_{m=1}^{M}\mathbb{1}_{par(X_i^m)=j_i,X_i^m=k_i}|^oX,\Theta^r) \\
&= \sum_{i,j_i,k_i} log(\theta_{i,j_i,k_i})\mathbb{E}\sum_{m=1}^{M}(\mathbb{1}_{par(X_i^m)=j_i,X_i^m=k_i}|^oX,\Theta^r) \\
&= \sum_{i,j_i,k_i} log(\theta_{i,j_i,k_i})\sum_{m=1}^{M}\mathbb{E}(\mathbb{1}_{par(X_i^m)=j_i,X_i^m=k_i}|^oX,\Theta^r) \\
&= \sum_{i,j_i,k_i} log(\theta_{i,j_i,k_i})\sum_{m=1}^{M}P(par(X_i^m)=j_i,X_i^m=k_i|^oX,\Theta^r) \\
&= \sum_{i,j_i,k_i} log(\theta_{i,j_i,k_i})\sum_{m=1}^{M}\gamma_{i,j_i,k_i} \\
&= \sum_{i,j_i,k_i} log(\theta_{i,j_i,k_i})\widehat{N_{i,j_i,k_i}}
\end{aligned}
\tag{5.20}
$$

Ceci est possible grâce à la propriété disant que l'espérance de la fonction indicatrice d'un évènement est égale à la probabilité de cet évènement.

La phase *Expectation* consiste donc à calculer $\widehat{N_{i,j_i,k_i}}$ par inférence, puis à en déduire $Q(\Theta, \Theta^r)$. La phase *Maximization* consiste à maximiser $\sum_{k_i} log(\theta_{i,j_i,k_i})\widehat{N_{i,j_i,k_i}}$ sous les contraintes $\sum_{k_i} \theta_{i,j_i,k_i} = 1$ et $0 \leq \theta_{i,j_i,k_i} \leq 1$, pour toutes valeurs du triplet (i, j_i, k_i).

À l'aide de la méthode des multiplicateurs de Lagrange, on obtient l'équation suivante de mise à jour du jeu de paramètres Θ :

$$\Theta^{r+1}_{i,j_i,k_i} = \frac{\widehat{N_{i,j_i,k_i}}}{\sum_{k_i} \widehat{N_{i,j_i,k_i}}} \tag{5.21}$$

Annexe 6 : Inférence exacte dans les réseaux Bayésiens dynamiques avec l'algorithme d'interface

Nous allons dérouler l'algorithme d'interface pour une application numérique basée sur l'exemple présenté dans la section 3.2.2. On reprendra tous les éléments présentés à cet endroit ; il est donc indispensable de les avoir vus pour comprendre ce qui va suivre. Comme dans l'exemple de l'Annexe 6, toutes nos variables sont binaires, B, C, et F sont cachées, et A, D, et G sont observées, comme illustré sur le schéma 5.29. On considère un réseau à seulement deux tranches pour réduire les calculs.

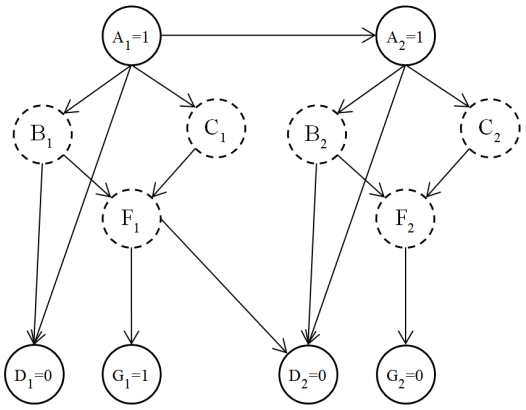


FIGURE 5.29 – Schéma du réseau bayésien dynamique utilisé pour l'exemple d'algorithme interface

Nous définissons les probabilités a priori et conditionnelles suivantes :

i	$P(A_1 = i)$
0	$P(\bar{A_1}) = 0.7$
1	$P(A_1) = 0.3$

	$P(A_t = i \mid A_{t-1} = j)$	
j \ i	0	1
0	$P(\bar{A_t} \mid \bar{A_{t-1}}) = 0.7$	$P(A_t \mid \bar{A_{t-1}}) = 0.3$
1	$P(\bar{A_t} \mid A_{t-1}) = 0.3$	$P(A_t \mid A_{t-1}) = 0.7$

$P(B_t = i \mid A_t = j)$		
j \ i	0	1
0	$P(\bar{B}_t \mid \bar{A}_t) = 0.2$	$P(B_t \mid \bar{A}_t) = 0.8$
1	$P(\bar{B}_t \mid A_t) = 0.4$	$P(B_t \mid A_t) = 0.6$

$P(C_t = i \mid A_t = j)$		
j \ i	0	1
0	$P(\bar{C}_t \mid \bar{A}_t) = 0.9$	$P(C_t \mid \bar{A}_t) = 0.1$
1	$P(\bar{C}_t \mid A_t) = 0.5$	$P(C_t \mid A_t) = 0.5$

$P(D_1 = i \mid A_1 = j, B_1 = k)$		
j,k \ i	0	1
00	$P(\bar{D}_1 \mid \bar{A}_1, \bar{B}_1) = 0.11$	$P(D_1 \mid \bar{A}_1, \bar{B}_1) = 0.89$
01	$P(\bar{D}_1 \mid \bar{A}_1, B_1) = 0.22$	$P(D_1 \mid \bar{A}_1, B_1) = 0.78$
10	$P(\bar{D}_1 \mid A_1, \bar{B}_1) = 0.33$	$P(D_1 \mid A_1, \bar{B}_1) = 0.67$
11	$P(\bar{D}_1 \mid A_1, B_1) = 0.44$	$P(D_1 \mid A_1, B_1) = 0.56$

$P(D_t = i \mid A_t = j, B_t = k, F_{t-1} = l)$		
j,k,l \ i	0	1
000	$P(\bar{D}_t \mid \bar{A}_t, \bar{B}_t, \bar{F}_{t-1}) = 0.11$	$P(D_t \mid \bar{A}_t, \bar{B}_t, \bar{F}_{t-1}) = 0.89$
001	$P(\bar{D}_t \mid \bar{A}_t, \bar{B}_t, F_{t-1}) = 0.22$	$P(D_t \mid \bar{A}_t, \bar{B}_t, F_{t-1}) = 0.78$
010	$P(\bar{D}_t \mid \bar{A}_t, B_t, \bar{F}_{t-1}) = 0.56$	$P(D_t \mid \bar{A}_t, B_t, \bar{F}_{t-1}) = 0.44$
011	$P(\bar{D}_t \mid \bar{A}_t, B_t, F_{t-1}) = 0.67$	$P(D_t \mid \bar{A}_t, B_t, F_{t-1}) = 0.33$
100	$P(\bar{D}_t \mid A_t, \bar{B}_t, \bar{F}_{t-1}) = 0.33$	$P(D_t \mid A_t, \bar{B}_t, \bar{F}_{t-1}) = 0.67$
101	$P(\bar{D}_t \mid A_t, \bar{B}_t, F_{t-1}) = 0.44$	$P(D_t \mid A_t, \bar{B}_t, F_{t-1}) = 0.56$
110	$P(\bar{D}_t \mid A_t, B_t, \bar{F}_{t-1}) = 0.78$	$P(D_t \mid A_t, B_t, \bar{F}_{t-1}) = 0.22$
111	$P(\bar{D}_t \mid A_t, B_t, F_{t-1}) = 0.89$	$P(D_t \mid A_t, B_t, F_{t-1}) = 0.11$

$P(F_t = i \mid B_t = j, C_t = k)$		
j,k \ i	0	1
00	$P(\bar{F}_t \mid \bar{B}_t, \bar{C}_t) = 0.05$	$P(F_t \mid \bar{B}_t, \bar{C}_t) = 0.95$
01	$P(\bar{F}_t \mid \bar{B}_t, C_t) = 0.15$	$P(F_t \mid \bar{B}_t, C_t) = 0.85$
10	$P(\bar{F}_t \mid B_t, \bar{C}_t) = 0.25$	$P(F_t \mid B_t, \bar{C}_t) = 0.75$
11	$P(\bar{F}_t \mid B_t, C_t) = 0.35$	$P(F_t \mid B_t, C_t) = 0.65$

$P(G_t = i \mid F_t = j)$		
j \ i	0	1
0	$P(\bar{G}_t \mid \bar{F}_t) = 0.55$	$P(G_t \mid \bar{F}_t) = 0.45$
1	$P(\bar{G}_t \mid F_t) = 0.66$	$P(G_t \mid F_t) = 0.34$

Pour les variables appartenant à l'interface *Forward*, on distingue deux cas ; lorsque l'on est à l'instant initial $t = 1$, et quand on se situe à un instant quelconque $t > 1$. On a donc défini une loi de probabilité pour chacun.

La répartition des distributions des variables dans les différentes cliques est présentée sur le schéma 5.30.

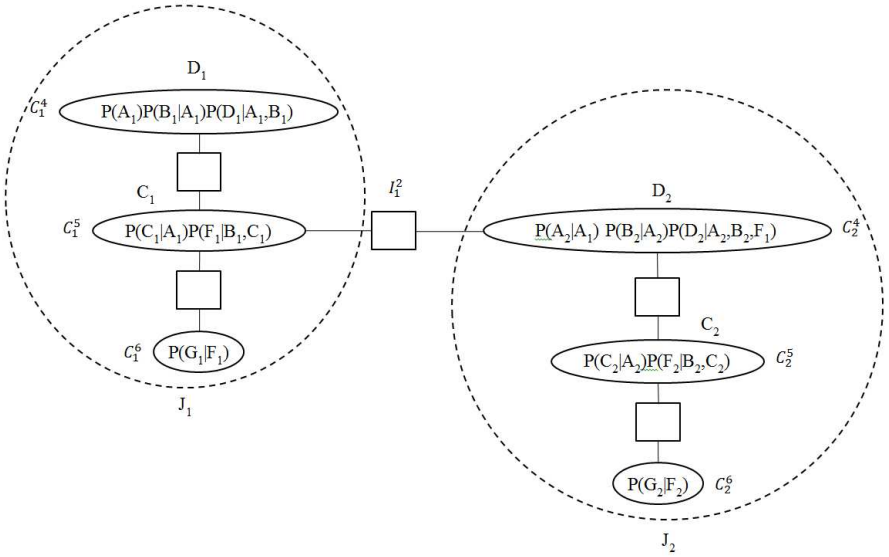


FIGURE 5.30 – Affectation des potentiels initiaux des cliques et des séparateurs

À l'instant t=1, la variable A est observée à 1, la variable D à 0, et la variable G à 1. À l'instant t=2, la variable A est observée à 1, la variable D à 0, et la variable G à 0. Par conséquent, la probabilité que ces variables ne valent pas la valeur observée est de 0. Les autres valeurs sont laissées telles quelles. On initialise donc les potentiels des cliques comme suit :

$\psi^0_{C_1^4} = P(A_1 = i)P(B_1 = j\|A_1 = i)P(D_1 = k\|A_1 = i, B_1 = j)$		
j,k \ i	0	1
00	0	0.0396
01	0	0
10	0	0.0792
11	0	0

$\psi^0_{C_1^5} = P(C_1 = i\|A_1 = j)P(F_1 = k\|B_1 = l, C_1 = i)$				
k,l \ i,j	00	01	10	11
00	0	0.025	0	0.075
01	0	0.125	0	0.175
10	0	0.475	0	0.425
11	0	0.375	0	0.325

$\psi^0_{C_1^6} = P(G_1 = i\|F_1 = j)$		
j \ i	0	1
0	0	0.45
1	0	0.34

$\psi^0_{C_2^4} = P(A_2 = i\|A_1 = j)P(B_2 = k\|A_2 = i)P(D_2 = l\|A_2 = i, B_2 = k, F_1 = m)$				
k,l,m \ i,j	00	01	10	11
000	0	0	0	0.0924
001	0	0	0	0.1232
010	0	0	0	0
011	0	0	0	0
100	0	0	0	0.3276
101	0	0	0	0.3738
110	0	0	0	0
111	0	0	0	0

$\psi^0_{C_2^5} = P(C_2 = i\|A_2 = j)P(F_2 = k\|B_2 = l, C_2 = i)$				
k,l \ i,j	00	01	10	11
00	0	0.025	0	0.075
01	0	0.125	0	0.175
10	0	0.475	0	0.425
11	0	0.375	0	0.325

$\psi^0_{C_2^6} = P(G_2 = i\|F_2 = j)$		
j \ i	0	1
0	0.55	0
1	0.66	0

Nous allons maintenant dérouler l'algorithme de *Message Passing* :
Passe *forward*, itération 1 :

$\psi^f_{C_1^4} = 1 \times \psi^0_{C_1^4}$		
B_1, D_1 \ A_1	0	1
00	0	0.0396
01	0	0
10	0	0.0792
11	0	0

$\mu_{C_1^4 \to C_1^5} = \sum_{A_1 B_1 D_1 \setminus A_1 B_1} \psi^f_{C_1^4} = \sum_{D_1} \psi^f_{C_1^4}$		
B_1 \ A_1	0	1
0	0	0.0396
1	0	0.0792

$\psi^f_{C_1^6} = 1 \times \psi^0_{C_1^6}$		
F_1 \ G_1	0	1
0	0	0.45
1	0	0.34

F_1	$\mu_{C_1^6 \to C_1^5} = \sum_{F_1 G_1 \setminus F_1} \psi^f_{C_1^6} = \sum_{G_1} \psi^f_{C_1^6}$
0	0.45
1	0.34

$\psi^f_{C_1^5} = \mu_{C_1^4 \to C_1^5} \psi^0_{C_1^5} \mu_{C_1^6 \to C_1^5}$				
F_1, B_1 \ C_1, A_1	00	01	10	11
00	0	0.0004455	0	0.0013365
01	0	0.004455	0	0.006237
10	0	0.0063954	0	0.0057222
11	0	0.010098	0	0.0087516

Passe *Forward*, itération 2 :

$\psi(I_1^2 \mid A_1 = 1, D_1 = 0, G_1 = 1) = \sum_{A_1 B_1 C_1 F_1 \setminus A_1 F_1} \psi^f_{C_1^5} = \sum_{B_1 C_1} \psi^f_{C_1^5}$		
F_1 \ A_1	0	1
0	0	0.012474
1	0	0.0309672

$\psi^0_{C_2^4} = \psi^0_{C_2^4} \psi(I_1^2 \mid A_1 = 1, D_1 = 0, G_1 = 1)$				
B_2, D_2, F_1 \ A_2, A_1	00	01	10	11
000	0	0	0	0.0011525976
001	0	0	0	0.00381515904
010	0	0	0	0
011	0	0	0	0
100	0	0	0	0.0040864824
101	0	0	0	0.01157553936
110	0	0	0	0
111	0	0	0	0

$\psi^f_{C_2^4} = 1 \times \psi^0_{C_2^4}$				
B_2, D_2, F_1 \ A_2, A_1	00	01	10	11
000	0	0	0	0.0011525976
001	0	0	0	0.00381515904
010	0	0	0	0
011	0	0	0	0
100	0	0	0	0.0040864824
101	0	0	0	0.01157553936
110	0	0	0	0
111	0	0	0	0

$\mu_{C_2^4 \to C_2^5} = \sum_{A_2 B_2 D_2 A_1 F_1 \setminus A_2 B_2} \psi^f_{C_2^4} = \sum_{D_2 A_1 F_1} \psi^f_{C_2^4}$		
B_2 \ A_2	0	1
0	0	0.00496775664
1	0	0.01566202176

$\psi^f_{C_2^6} = 1 \times \psi^0_{C_2^6}$		
F_2 \ G_2	0	1
0	0.55	0
1	0.66	0

F_2	$\mu_{C_2^6 \to C_2^5} = \sum_{F_2 G_2 \setminus F_2} \psi^f_{C_2^6} = \sum_{G_2} \psi^f_{C_2^6}$
0	0.55
1	0.66

$\psi^f_{C_2^5} = \mu_{C_2^4 \to C_2^5} \psi^0_{C_2^5} \mu_{C_2^6 \to C_2^5}$				
F_2, B_2 \ C_2, A_2	00	01	10	11
00	0	0.0000683066538	0	0.0002049199614
01	0	0.001076763996	0	0.0015074695944
10	0	0.00155739170664	0	0.00139345573752
11	0	0.0038763503856	0	0.00335950366752

Passe *Backward* (avec normalisation), itération 1 :

$\psi^b_{C_2^5} = 1 \times \psi^f_{C_2^5}$				
F_2, B_2 \ C_2, A_2	00	01	10	11
00	0	0.005236569	0	0.015709707
01	0	0.082547581	0	0.115566614
10	0	0.119393775	0	0.106826009
11	0	0.297171292	0	0.257548453

F_2	$\mu_{C_2^5 \to C_2^6} = \sum_{A_2 B_2 C_2 F_2 \setminus F_2} \frac{\psi^b_{C_2^5}}{\mu_{C_2^6 \to C_2^5}} = \sum_{A_2 B_2 C_2} \frac{\psi^b_{C_2^5}}{\mu_{C_2^6 \to C_2^5}}$
0	0.398291765
1	1.183241711

$\mu_{C_2^5 \to C_2^4} = \sum_{A_2 B_2 C_2 F_2 \setminus A_2 B_2} \frac{\psi^b_{C_2^5}}{\mu_{C_2^4 \to C_2^5}} = \sum_{C_2 F_2} \frac{\psi^b_{C_2^5}}{\mu_{C_2^4 \to C_2^5}}$		
B_2 \ A_2	0	1
0	0	49.75405969
1	0	48.0674814

$\psi^b_{C_2^6} = \mu_{C_2^5 \to C_2^6} \psi^f_{C_2^6}$		
F_2 \ G_2	0	1
0	0.219060471	0
1	0.780939529	0

$\psi^b_{C_2^4} = \mu_{C_2^5 \to C_2^4} \psi^f_{C_2^4}$				
B_2, D_2, F_1 \ A_2, A_1	00	01	10	11
000	0	0	0	0.05734641
001	0	0	0	0.189819651
010	0	0	0	0
011	0	0	0	0
100	0	0	0	0.196426917
101	0	0	0	0.556407023
110	0	0	0	0
111	0	0	0	0

Passe *Backward* (avec normalisation), itération 2 :

$\psi(I_1^2 \mid A_1=1, D_1=0, G_1=1, A_2=1, D_2=0, G_2=0) = \sum_{A_2B_2D_2A_1F_1\setminus A_1F_1} \psi^b_{C_2^4} = \sum_{A_2B_2D_2} \psi^b_{C_2^4}$		
F_1 \ A_1	0	1
0	0	0.253773327
1	0	0.746226673

$\psi^b_{C_1^5} = \psi^f_{C_1^5} \frac{\psi(I_1^2 \mid A_1=1, D_1=0, G_1=1, A_2=1, D_2=0, G_2=0)}{\psi(I_1^2 \mid A_1=1, D_1=0, G_1=1)}$				
F_1, B_1 \ C_1, A_1	00	01	10	11
00	0	0.009063333	0	0.027189999
01	0	0.090633331	0	0.126886663
10	0	0.15411203	0	0.137889711
11	0	0.243334785	0	0.210890147

F_1	$\mu_{C_1^5\to C_1^6} = \sum_{A_1B_1C_1F_1\setminus F_1} \frac{\psi^b_{C_1^5}}{\mu_{C_1^6\to C_1^5}} = \sum_{A_1B_1C_1} \psi^b_{C_1^5}$
0	0.563940726
1	2.194784334

$\mu_{C_1^5\to C_1^4} = \sum_{A_1B_1C_1F_1\setminus A_1B_1} \frac{\psi^b_{C_1^5}}{\mu_{C_1^4\to C_1^5}} = \sum_{C_1F_1} \psi^b_{C_1^5}$		
B_1 \ A_1	0	1
0	0	8.28926955
1	0	8.481627852

$\psi^b_{C_1^6} = \mu_{C_1^5\to C_1^6}\psi^f_{C_1^6}$		
F_1 \ G_1	0	1
0	0	0.253773327
1	0	0.746226673

$\psi^b_{C_1^4} = \mu_{C_1^5\to C_1^4}\psi^f_{C_1^4}$		
A_1, B_1 \ D_1	0	1
00	0	0
01	0	0
10	00328255074	0
11	0.671744926	0

Supposons maintenant que l'on souhaite obtenir les probabilités marginales de toutes les variables du réseau à chaque instant. On peut les calculer à partir des potentiels finaux de chaque clique, en marginalisant sur la variable souhaitée n'importe quelle clique la contenant. On obtient les résultats suivants :

i	$P(A_1 = i)$
0	$P(\bar{A_1}) = 0$
1	$P(A_1) = 1$

i	$P(B_1 = i)$
0	$P(\bar{B_1}) = 0.328255074$
1	$P(B_1) = 0.671744926$

i	$P(C_1 = i)$
0	$P(\bar{C_1}) = 0.49714348$
1	$P(C_1) = 0.50285652$

i	$P(D_1 = i)$
0	$P(\bar{D_1}) = 1$
1	$P(D_1) = 0$

i	$P(F_1 = i)$
0	$P(\bar{F_1}) = 0.253773327$
1	$P(F_1) = 0.746226673$

i	$P(G_1 = i)$
0	$P(\bar{G_1}) = 0$
1	$P(G_1) = 1$

i	$P(A_2 = i)$
0	$P(\bar{A_2}) = 0$
1	$P(A_2) = 1$

i	$P(B_2 = i)$
0	$P(\bar{B_2}) = 0.24716606$
1	$P(B_2) = 0.75283394$

i	$P(C_2 = i)$
0	$P(\bar{C_2}) = 0.504349217$
1	$P(C_2) = 0.495650783$

i	$P(D_2 = i)$
0	$P(\bar{D_2}) = 1$
1	$P(D_2) = 0$

i	$P(F_2 = i)$
0	$P(\bar{F_2}) = 0.219060471$
1	$P(F_2) = 0.780939529$

i	$P(G_2 = i)$
0	$P(\bar{G_2}) = 1$
1	$P(G_2) = 0$

Annexe 7 : Détails sur l'obtention des formules de l'EM appliqué aux réseaux Bayésiens dynamiques

Nous représenterons un paramètre du réseau sachant ses parents par $\theta_{i,l_i,j_{l_i},k_i} = P(X_i = k_i | \{par(X_i)/ordre(par(X_i)) \leq ordres\{par(X_i)\}_{l_i}\} = j_{l_i})$, avec l_i la tranche de définition du nœud X_i, $ordre(par(X_i))$ l'ordre d'un parent, et $ordres\{par(X_i)\} = \{ordre(par(X_i))/\forall p, q \in |ordres\{par(X_i)\}|/p < q, ordre(par(X_i))_p < ordre(par(X_i))_q\}$ l'ensemble des ordres uniques des parents classés en ordre croissant. Si un nœud se trouve dans une tranche temporelle t ($t \geq 0$), alors un parent d'ordre i ($i \geq 0$) de ce nœud se trouve dans la tranche temporelle $t-i$.

L'ensemble des nœuds du réseau bayésien dynamique comportant N variables et T tranches temporelles ($T > max(ordres\{par(X_i)\})$) est représenté par $X = \begin{pmatrix} X_1^1 & ... & X_N^1 \\ ... & ... & ... \\ X_1^T & ... & X_N^T \end{pmatrix}$

On considère maintenant l'ensemble $Ordres\{par(X_i)\} = ordres\{par(X_i)\} + \{0\} + \{T\}$, l'ensemble des ordres uniques des parents auquel on ajoute 0 et T afin de s'assurer que l'excursion de l'ensemble correspond bien aux tranches temporelles existant dans le réseau. On définit également $par(X_i^t)_{l_i} = \{par(X_i^t)/ordre(par(X_i^t)) \leq Ordres\{par(X_i)\}_{l_i}\}$, la combinaison de parents du nœud X_i^t à la tranche de définition l_i.

La probabilité conjointe (ou vraisemblance) de tous les nœuds de ce réseau est :

$$\begin{aligned} P(X;\Theta) &= \prod_{l_i=1}^{|Ordres\{par(X_i)\}|-1} \prod_{t=Ordres\{par(X_i)\}_{l_i}}^{Ordres\{par(X_i)\}_{l_i+1}-1} \prod_{i=1}^{N} P(X_i^t | par(X_i^t)_{l_i}) \\ &= \prod_{l_i=1}^{|Ordres\{par(X_i)\}|-1} \prod_{t=Ordres\{par(X_i)\}_{l_i}}^{Ordres\{par(X_i)\}_{l_i+1}-1} \prod_{i=1}^{N} \theta_{i,l_i,par(X_i^t)_{l_i},X_i^t} \end{aligned} \tag{5.22}$$

où Θ représente l'ensemble des paramètres du réseau. Pour éviter l'underflow lors de l'implémentation pratique de l'algorithme, on utilisera la log-vraisemblance, définie comme suit :

$$L(X;\Theta) = log(P(X;\Theta)) = log(\prod_{l_i=1}^{|Ordres\{par(X_i)\}|-1} \prod_{t=Ordres\{par(X_i)\}_{l_i}}^{Ordres\{par(X_i)\}_{l_i+1}-1} \prod_{i=1}^{N} \theta_{i,l_i,par(X_i^t)_{l_i},X_i^t}) \tag{5.23}$$

$$= \sum_{l_i=1}^{|Ordres\{par(X_i)\}|-1} \sum_{t=Ordres\{par(X_i)\}_{l_i}}^{Ordres\{par(X_i)\}_{l_i+1}-1} \sum_{i=1}^{N} log(\theta_{i,l_i,par(X_i^t)_{l_i},X_i^t})$$

On considère maintenant que l'on dispose d'un ensemble de M réalisations indépendantes de notre réseau bayésien dynamique. On définit $Ordres\{par(X_i)\}^m = ordres\{par(X_i)\} + \{0\} + \{T_m\}$, car le nombre de total de tranches temporelles est potentiellement dépendant de la réalisation. L'ensemble de ces réalisations est représenté par :

$$D = \begin{pmatrix} {}_1X_1^1 & ... & {}_1X_N^1 \\ ... & ... & ... \\ {}_1X_1^{T_1} & ... & {}_1X_N^{T_1} \end{pmatrix} ... \begin{pmatrix} {}_MX_1^1 & ... & {}_MX_N^1 \\ ... & ... & ... \\ {}_MX_1^{T_M} & ... & {}_MX_N^{T_M} \end{pmatrix} \tag{5.24}$$

Grâce à l'indépendance des réalisations, la probabilité conjointe (ou vraisemblance) se décompose en un produit :

$$P(D;\Theta) = \prod_{m=1}^{M} p({}_mX;\Theta) = \prod_{m=1}^{M} \prod_{l_i=1}^{|Ordres\{par(X_i)\}^m|-1} \prod_{t=Ordres\{par(X_i)\}_{l_i}^m}^{Ordres\{par(X_i)\}_{l_i+1}^m-1} \prod_{i=1}^{N} \theta_{i,l_i,par(X_i^t)_{l_i},mX_i^t} \tag{5.25}$$

On obtient ensuite la log-vraisemblance suivante :

$$L(D;\Theta) = log(P(D;\Theta)) \tag{5.26}$$

$$= log(\prod_{m=1}^{M} \prod_{l_i=1}^{|Ordres\{par(X_i)\}^m|-1} \prod_{t=Ordres\{par(X_i)\}_{l_i}^m}^{Ordres\{par(X_i)\}_{l_i+1}^m-1} \prod_{i=1}^{N} \theta_{i,l_i,par(X_i^t)_{l_i},mX_i^t})$$

$$= \sum_{m=1}^{M} \sum_{l_i=1}^{|Ordres\{par(X_i)\}^m|-1} \sum_{t=Ordres\{par(X_i)\}_{l_i}^m}^{Ordres\{par(X_i)\}_{l_i+1}^m-1} \sum_{i=1}^{N} log(\theta_{i,l_i,par(X_i^t)_{l_i},mX_i^t})$$

$$= \sum_{m=1}^{M} \sum_{l_i=1}^{|Ordres\{par(X_i)\}^m|-1} \sum_{t=Ordres\{par(X_i)\}_{l_i}^m}^{Ordres\{par(X_i)\}_{l_i+1}^m-1} \sum_{i=1}^{N} log(\prod_{j_{l_i},k_i} \theta_{i,l_i,j_{l_i},k_i}^{\mathbb{1}_{par(X_i^t)_{l_i}=j_{l_i},mX_i^t=k_i}})$$

$$= \sum_{m=1}^{M} \sum_{l_i=1}^{|Ordres\{par(X_i)\}^m|-1} \sum_{t=Ordres\{par(X_i)\}_{l_i}^m}^{Ordres\{par(X_i)\}_{l_i+1}^m-1} \sum_{i=1}^{N} \sum_{j_{l_i},k_i} \mathbb{1}_{par(X_i^t)_{l_i}=j_{l_i},mX_i^t=k_i} log(\theta_{i,l_i,j_{l_i},k_i})$$

$$= \sum_{i,l_i,j_{l_i},k_i} log(\theta_{i,l_i,j_{l_i},k_i}) \sum_{m=1}^{M} \sum_{t=Ordres\{par(X_i)\}_{l_i}^m}^{Ordres\{par(X_i)\}_{l_i+1}^m-1} \mathbb{1}_{par(X_i^t)_{l_i}=j_{l_i},mX_i^t=k_i}$$

$$= \sum_{i,l_i,j_{l_i},k_i} log(\theta_{i,l_i,j_{l_i},k_i}) N_{i,l_i,j_{l_i},k_i}$$

N_{i,l_i,j_{l_i},k_i} représente le nombre de fois où la variable X_i vaut k_i et ses parents de la tranche de définition l_i valent j_{l_i}. Une partie des nœuds n'étant pas observée, N_{i,l_i,j_{l_i},k_i} n'est pas calculable directement.

La fonction Q de l'algorithme EM s'écrit donc :

$$
\begin{aligned}
Q(\Theta,\Theta^r) &= \mathbb{E}(L(X;\Theta)|^oX,\Theta^r) \qquad (5.27)\\
&= \mathbb{E}(\sum_{i,l_i,j_{l_i},k_i} log(\theta_{i,l_i,j_{l_i},k_i}) \sum_{m=1}^{M} \sum_{t=Ordres\{par(X_i)\}_{l_i}^{m}}^{Ordres\{par(X_i)\}_{l_i+1}^{m}-1} \mathbb{1}_{par(X_i^t)_{l_i}=j_{l_i},_m X_i^t=k_i}|^oX,\Theta^r)\\
&= \sum_{i,l_i,j_{l_i},k_i} log(\theta_{i,l_i,j_{l_i},k_i}) \mathbb{E}(\sum_{m=1}^{M} \sum_{t=Ordres\{par(X_i)\}_{l_i}^{m}}^{Ordres\{par(X_i)\}_{l_i+1}^{m}-1} \mathbb{1}_{par(X_i^t)_{l_i}=j_{l_i},_m X_i^t=k_i}|^oX,\Theta^r)\\
&= \sum_{i,l_i,j_{l_i},k_i} log(\theta_{i,l_i,j_{l_i},k_i}) \sum_{m=1}^{M} \sum_{t=Ordres\{par(X_i)\}_{l_i}^{m}}^{Ordres\{par(X_i)\}_{l_i+1}^{m}-1} \mathbb{E}(\mathbb{1}_{par(X_i^t)_{l_i}=j_{l_i},_m X_i^t=k_i}|^oX,\Theta^r)\\
&= \sum_{i,l_i,j_{l_i},k_i} log(\theta_{i,l_i,j_{l_i},k_i}) \sum_{m=1}^{M} \sum_{t=Ordres\{par(X_i)\}_{l_i}^{m}}^{Ordres\{par(X_i)\}_{l_i+1}^{m}-1} P(par(X_i^t)_{l_i}=j_{l_i},_m X_i^t=k_i|^oX,\Theta^r)\\
&= \sum_{i,l_i,j_{l_i},k_i} log(\theta_{i,l_i,j_{l_i},k_i}) \sum_{m=1}^{M} \sum_{t=Ordres\{par(X_i)\}_{l_i}^{m}}^{Ordres\{par(X_i)\}_{l_i+1}^{m}-1} \gamma_{i,l_i,j_{l_i},k_i}\\
&= \sum_{i,l_i,j_{l_i},k_i} log(\theta_{i,l_i,j_{l_i},k_i}) \widehat{N_{i,l_i,j_{l_i},k_i}}
\end{aligned}
$$

Ceci est possible grâce à la propriété disant que l'espérance de la fonction indicatrice d'un évènement est égale à la probabilité de cet évènement.

La phase *Expectation* consiste donc à calculer $\widehat{N_{i,l_i,j_{l_i},k_i}}$ par inférence, puis à en déduire $Q(\Theta,\Theta^r)$. La phase *Maximization* consiste à maximiser $\sum_{k_i} log(\theta_{i,l_i,j_{l_i},k_i})\widehat{N_{i,l_i,j_{l_i},k_i}}$ sous les contraintes $\sum_{k_i} \theta_{i,l_i,j_{l_i},k_i} = 1$ et $0 \leq \theta_{i,l_i,j_{l_i},k_i} \leq 1$, pour toutes valeurs du quadruplet (i,l_i,j_{l_i},k_i).

À l'aide de la méthode des multiplicateurs de Lagrange, on obtient l'équation suivante de mise à jour du jeu de paramètres Θ :

$$\Theta_{i,l_i,j_{l_i},k_i}^{r+1} = \frac{\widehat{N_{i,l_i,j_{l_i},k_i}}}{\sum_{k_i} \widehat{N_{i,l_i,j_{l_i},k_i}}} \qquad (5.28)$$

Bibliographie

[1] L. Figuier, *Les Merveilles de la Science*.

[2] Wikipedia. Origine des télécommunications. [Online]. Available : https://fr.wikipedia.org/wiki/Histoire_des_tÃľlÃľcommunications

[3] K. Musem. Les télécommunications dans la grèce antique - introduction. [Online]. Available : http://kotsanas.com/fr/exh.php?exhibit=1201000

[4] E. Moulinsard. Invention de l'électricité (1800). [Online]. Available : https://www.histoire-pour-tous.fr/inventions/309-invention-electricite.html

[5] A. J. Coly. De la 0g à la 4g, soixante ans de progrès. [Online]. Available : https://www.agenceecofin.com/mobile/0101-2732-de-la-0g-a-la-4g-soixante-ans-de-progres

[6] routeur 5G.fr. Réseaux mobiles 1g, 2g, 3g, 4g et 5g : l'essentiel à savoir. [Online]. Available : https://routeur-5g.fr/guide-dachat/comparatif-entre-3g-4g-et-5g/

[7] G. de la Banque Mondiale. Abonnements à la téléphonie mobile (pour 100 habitants) - france. [Online]. Available : https://donnees.banquemondiale.org/indicateur/IT.CEL.SETS.P2?locations=FR

[8] objetconnecte.net. Objets connectés : Histoire et définitions. [Online]. Available : https://objetconnecte.net/histoire-definitions-objet-connecte/

[9] Antoine. Marché de l'iot en france et dans le monde, chiffres importants à connaître. [Online]. Available : https://www.carnetdebord.info/marche-iot-en-france/

[10] Georges. 5g et iot : 7 chiffres à retenir en 2021. [Online]. Available : https://www.matooma.com/fr/s-informer/actualites-iot-m2m/5g-iot

[11] T. Salman and R. Jain, "A survey of protocols and standards for internet of things," *ArXiv*, vol. abs/1903.11549, 2019.

[12] H. Zimmermann, "Osi reference model - the iso model of architecture for open systems interconnection," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, 1980.

[13] U. User, F. Touati, A. Ben Mnaouer, and A. Ghaleb, "A comparative analysis of ble and ieee802.15.4 (6lowpan) for u-healthcare applications," 11 2013.

[14] N. Q. Mehmood and R. Culmone, "An ant+ protocol based health care system," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, 2015, pp. 193–198.

[15] A. Lothoré, "Protocoles de communication, frameworks et systèmes d'exploitation pour les objets connectés," Web White Paper on Linux Embedded, March 2016. [Online]. Available : http://www.linuxembedded.fr/2016/03/

protocoles-de-communication-frameworks-et-systemes-dexploitation-pour-les-objets-connectes/#80211

[16] *DASH7 Alliance Wireless Sensor and Actuator Network Protocol VERSION 1.1*, DASH7 Alliance Std.

[17] S. Oudji, "Analyse de la robustesse et des améliorations potentielles du protocole radio-fréquences sub-ghz knx utilisé pour l'iot domotique," Ph.D. dissertation, UNIVERSITE DE LIMOGES, 2016.

[18] M. S. Nixon, W. A. Louis, and H. Blvd, "A comparison of wirelesshart™ and isa100.11a," 2012.

[19] S. Marksteiner, V. J. Exposito Jimenez, H. Valiant, and H. Zeiner, "An overview of wireless iot protocol security in the smart home domain," in *2017 Internet of Things Business Models, Users, and Networks*, 2017, pp. 1–8.

[20] Y. Yang. (2008) Microchip miw p2p wireless protocol. Microchip.

[21] Waviot, "What is nb-fi protocol," online presentation on waviot website.

[22] "Le protocole rubee devient un standard ieee pour la transmission radio," Web article on Mesures, March 2009. [Online]. Available : http://www.mesures.com/archives/item/3432-Le-protocole-RuBee-devient-un-standard-IEEE-pour-la-transmission-radio

[23] *Thread Stack Fundamentals*, Thread Group Std.

[24] H. Boeglen, "The wave communications stack : Ieee 802.11p, 1609.4 and, 1609.3," TechnoCom presentation, September 2007.

[25] N. Docq, L. D. Rigaud, and M. Hababou, "La norme 802.16," Ingénieurs2000, Tech. Rep., 2003.

[26] V. Mohan, "An introduction to wireless m-bus."

[27] H. Harada, K. Mizutani, J. FUJIWARA, K. MOCHIZUKI, K. OBATA, and R. Okumura, "Ieee 802.15.4g based wi-sun communication systems," *IEICE Transactions on Communications*, vol. E100.B, 01 2017.

[28] C. Duvallet, "Les réseaux sans fils," Université du Havre CNAM SMB 214-215 lecture.

[29] C. Perkins, "Tutorial presentation for ietf 95," Tutorial, April 2016.

[30] R. International, "The future of uwb technology and ieee 802.15.4z," Web article on RealTrac website, February 2019. [Online]. Available : https://real-trac.com/en/company/blog/the-future-of-uwb-technology-and-the-ieee-802-15-4-z/

[31] K. L. Lueth, "State of the iot 2018 : Number of iot devices now at 7b - market accelerating," Web article on IoT Analytics, August 2018. [Online]. Available : https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/

[32] S. Labs, "The wireless protocols tying together the internet of things," White Paper, July 2018.

[33] U. Noreen, A. Bounceur, and L. Clavier, "Modeling interference for wireless sensor network simulators," 07 2017, pp. 1–6.

[34] Z. Alliance, "Zigbee alliance and thread group achieve a major milestone for iot interoperability with release of dotdot over thread specification and streamlined certification programs," Web article on Zigbee Alliance website, January 2019. [Online]. Available : https://www.zigbee.org/zigbee-alliance-and-thread-group-achieve-a-major-milestone-for-iot-interoperability-with-release-of-dotd

[35] H. Cruz-S'nchez, L. Havet, M. Chehaider, and Y.-Q. Song, "Mpigate : A solution to use heterogeneous networks for assisted living applications," in *2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing*, 2012, pp. 104–111.

[36] E. Hjelmvik and W. John, "Statistical protocol identification with spid : preliminary results," 01 2009.

[37] A. Li, C. Dong, S. Tang, F. wu, C. Tian, B. Tao, and H. Wang, "Demodulation-free protocol identification in heterogeneous wireless networks," *Computer Communications*, vol. 55, 09 2014.

[38] S. Hu, Y.-d. Yao, and Z. Yang, "Mac protocol identification using support vector machines for cognitive radio networks," *IEEE Wireless Communications*, vol. 21, no. 1, pp. 52–60, 2014.

[39] C. Zevala, "Iot memory : An overview of the options," Web article in Jaxenter, January 207. [Online]. Available : https://jaxenter.com/iot-memory-overview-options-131270.html

[40] ST, "Ultra-low-power arm cortex-m33 32-bit mcu+trustzone+fpu, 165 dmips, up to 512 kb flash memory, 256 kb sram, smps," Microcontroller datasheet, June 2019.

[41] T. Zimmermann, J. Hiller, H. Reelfs, P. Hein, and K. Wehrle, "Split : Smart protocol loading for the iot," in *EWSN*, 2018.

[42] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma, "Fastgrnn : A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network," 2019.

[43] J. Tang, D. Sun, S. Liu, and J.-L. Gaudiot, "Enabling deep learning on iot devices," *Computer*, vol. 50, no. 10, pp. 92–96, 2017.

[44] E. Dukes, "The cost of iot sensors is dropping fast," Ioffice website post, September 2018.

[45] T. Parker, "Design trends : Batteries for iot," Avnet website post.

[46] L. Majer, J. Mihálov, V. Stopjaková, R. Záluský, J. Brenkus, and M. Uram, "An rf network combiner towards a wireless multi-communication system for smart households," *2014 International Conference on Applied Electronics*, pp. 193–196, 2014.

[47] R. Akeela and B. Dezfouli, "Software-defined radios : Architecture, state-of-the-art, and challenges," *Comput. Commun.*, vol. 128, pp. 106–125, 2018.

[48] L. Alaus, J. Palicot, C. Roland, Y. Louet, and D. Noguet, "Promising technique of parameterization for reconfigurable radio, the common operators technique : Fundamentals and examples," *Signal Processing Systems*, vol. 62, pp. 173–185, 02 2011.

[49] L. Hong, "Experience of iot transceiver with affordable software-defined radio platform," 06 2020.

[50] J. Liang, H. H. Chen, and S. C. Liew, "Design and implementation of time-sensitive wireless iot networks on software-defined radio," 06 2020.

[51] O. Esoul and N. Walkinshaw, "Finding clustering configurations to accurately infer packet structures from network data," 10 2016.

[52] B. Sija, Y.-H. Goo, K.-S. Shim, H. Hasanova, and M.-S. Kim, "A survey of automatic protocol reverse engineering approaches, methods, and tools on the inputs and outputs view," *Security and Communication Networks*, vol. 2018, pp. 1–17, 02 2018.

[53] S. Kleber, L. Maile, and F. Kargl, "Survey of protocol reverse engineering algorithms : Decomposition of tools for static traffic analysis," *IEEE Communications Surveys & Tutorials*, vol. 21, pp. 526–561, 2019.

[54] F. Delaveau and Y. Livran, "Radiosurveillance du spectre- analyse et identification des transmissions," *Techniques de l'ingénieur*, 2012.

[55] A. Darwiche, "Inference in bayesian networks : A historical perspective."

[56] A. Ohri. Introduction to bayesian belief networks in 2021. [Online]. Available : https://www.jigsawacademy.com/blogs/data-science/bayesian-belief-network#What-are-the-Bayesian-Networks-used-for?

[57] A. Trifilò, S. Burschka, and E. Biersack, "Traffic to protocol reverse engineering," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–8.

[58] Y. Wang, N. Zhang, Y.-m. Wu, B.-b. Su, and Y.-j. Liao, "Protocol formats reverse engineering based on association rules in wireless environment," in *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 2013, pp. 134–141.

[59] Y. Wang, X. Yun, M. Z. Shafiq, L. Wang, A. X. Liu, Z. Zhang, D. Yao, Y. Zhang, and L. Guo, "A semantics aware approach to automated reverse engineering unknown protocols," in *2012 20th IEEE International Conference on Network Protocols (ICNP)*, 2012, pp. 1–10.

[60] J. Cai, J.-Z. Luo, and F. Lei, "Analyzing network protocols of application layer using hidden semi-markov model," *Mathematical Problems in Engineering*, vol. 2016, pp. 1–14, 01 2016.

[61] A. Aho and M. Corasick, "Efficient string matching : An aid to bibliographic search," *Commun. ACM*, vol. 18, pp. 333–340, 06 1975.

[62] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, May 2003.

[63] G. Heinrich, "Parameter estimation for text analysis," University of Leipzig, Germany, Technical Note, 2008.

[64] *Zigbee Specification*, ZigBee Standards Organization Std.

[65] *IEEE Std 802.15.-2003*, IEEE Std.

[66] C. Alchiekh Haydar, "Trust-based recommender systems," Theses, Université de Lorraine, Sep. 2014. [Online]. Available : https://hal.univ-lorraine.fr/tel-01751172

[67] K. Murphy, "Dynamic bayesian networks : Representation, inference and learning," Ph.D. dissertation, 01 2002.

[68] S. Tembo, S. Vaton, J.-L. Courant, and S. Gosselin, "A tutorial on the em algorithm for bayesian networks : Application to self-diagnosis of gpon-ftth networks," 09 2016, pp. 369–376.

[69] F. Jensen, S. Lauritzen, and K. Olesen, "Bayesian updating in causal probabilistic networks by local computations," *Computational Statistics Quarterly*, vol. 4, pp. 269–282, 1990.

[70] Sample captures. [Online]. Available : https://wiki.wireshark.org/SampleCaptures

[71] Captures. [Online]. Available : https://www.cloudshark.org/captures/36822505d7ab

[72] Captures. [Online]. Available : https://www.cloudshark.org/captures/46a9a369e6a9

[73] M. Bazoge. Apple, amazon, google et zigbee lancent un nouveau standard pour la domotique. [Online]. Available : https://www.igen.fr/domotique/2019/12/apple-amazon-google-et-zigbee-lancent-un-nouveau-standard-pour-la-domotique-111939

[74] N. Furno. Domotique : ne dites plus chip et zigbee, mais matter et connectivity standards alliance. [Online]. Available : https://www.igen.fr/domotique/2021/05/domotique-ne-dites-plus-chip-et-zigbee-mais-matter-et-connectivity-standard-alliance-122435

Titre : Contribution à l'apprentissage non supervisé de protocoles pour la couche de Liaison de Données dans les systèmes communicants, à l'aide des Réseaux Bayésiens

Mots clés : Protocol Reverse Engineering, Latent Dirichlet Allocation, Internet des Objets, Réseaux Bayésiens, semi-chaînes de Markov cachées, protocoles binaires

Résumé : Le monde des télécommunications est en rapide développement, surtout dans le domaine de l'internet des objets; dans un tel contexte, il serait utile de pouvoir analyser n'importe quel protocole inconnu auquel on pourrait se trouver confronté. Dans ce but, l'obtention de la machine d'états et des formats de trames du protocole cible est indispensable. Ces deux éléments peuvent être extraits de traces réseaux et/ou traces d'exécution à l'aide de techniques de Protocol Reverse Engineering (PRE).

A l'aide de l'analyse des performances de trois algorithmes utilisés dans des systèmes de PRE, nous avons découvert le potentiel des modèles basés sur les réseaux Bayésiens. Nous avons ensuite développé Bayesian Network Frame Format Finder (BaNet3F), notre propre modèle d'apprentissage de format de trames basé sur les réseaux Bayésiens, et nous avons montré que ses performances sont nettement supérieures à celles de l'état de l'art. BaNet3F inclut également une version optimisée de l'algorithme de Viterbi, applicable à un réseau Bayésien quelconque, grâce à sa capacité à générer lui-même les frontières de Markov nécessaires.

Title : Contribution to unsupervised learning of Data Link layer protocols in communicating systems, using Bayesian Networks

Keywords : Protocol Reverse Engineering, Latent Dirichlet Allocation, Internet of Things, Bayesian networks, hidden semi-Markov models, binary protocols

Abstract : The world of telecommunications is rapidly developing, especially in the area of the Internet of Things; in such a context, it would be useful to be able to analyze any unknown protocol one might encounter. For this purpose, obtaining the state machine and frame formats of the target protocol is essential. These two elements can be extracted from network traces and/or execution traces using Protocol Reverse Engineering (PRE) techniques.

By analyzing the performance of three algorithms used in PRE systems, we discovered the potential of models based on Bayesian networks. We then developed Bayesian Network Frame Format Finder (BaNet3F), our own frame format learning model based on Bayesian networks, and showed that its performance is significantly better than the state of the art. BaNet3F also includes an optimized version of the Viterbi algorithm, applicable to any Bayesian network, thanks to its ability to generate the necessary Markov boundaries itself.