Approche Tenseur-Train pour l'inférence dans les modèles à blocs stochastiques, application à la caractérisation de la biodiversité

Mohamed Anwar Abouabdallah

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR
DE L'UNIVERSITÉ DE BORDEAUX

École doctorale Mathématiques et Informatique

Mathématiques appliquées et calcul scientifique

Par **Mohamed Anwar Abouabdallah**

Approche Tenseur-Train pour l'inférence dans les modèles à blocs stochastiques, application à la caractérisation de la biodiversité

Sous la direction de : **Olivier COULAUD**
Co-directrice : **Nathalie PEYRARD**

Soutenue le 02 février 2023

Membres du jury :

Mme Sophie DONNET	Chargée de recherche	INRAE	Rapportrice
M. Jean-René POIRIER	Maître de conférences	INP-ENSEEIHT	Rapporteur
Mme Agnès BOUCHEZ	Directrice de recherche	INRAE	Examinatrice
M. Pierre-Henri WUILLEMIN	Maître de conférences	Lip6	Examinateur
M. Olivier COULAUD	Directeur de recherche	INRIA	Directeur de thèse
Mme Nathalie PEYRARD	Directrice de recherche	INRAE	Directrice de thèse
M. Alain FRANC	Directeur de recherche	INRAE/INRIA	Invité

Approche Tenseur-Train pour l'inférence dans les modèles à blocs stochastiques, application à la caractérisation de la biodiversité

Résumé :

Le modèle à blocs stochastiques (SBM, Stochastic Block Model) est un modèle graphique particulier permettant de classer des individus sur la base de leurs distances deux à deux. Il construit des groupes d'individus partageant les mêmes profils de distance intra et inter groupe. L'estimation des paramètres d'un SBM se fait classiquement par l'algorithme EM (Expectation-Maximisation) qui à chaque itération demande le calcul des marginales unaires et binaires de la loi jointe conditionnelle des groupes des individus. Une manière de faire ce calcul est de passer par une approximation champ moyen du modèle où les marginales binaires sont calculées comme produit des marginales unaires. C'est ce qui est mis en œuvre dans l'algorithme Variational EM. Cet algorithme conduit à des estimateurs de bonne qualité, cependant, on peut espérer améliorer l'inférence des marginales binaires.
L'objectif de ces travaux de thèse est de proposer une approche plus précise pour le calcul des marginales binaires faisant appel à l'algèbre tensorielle, par une approche de type Tensor Train (TT), en étendant l'approche de Novikov & al, 2014 sur le calcul de la constante de normalisation d'un modèle graphique.
Dans un premier temps, afin de motiver ce travail méthodologique sur les SBM, nous avons réalisé un clustering de matrices de distances génétiques dans un échantillon de marqueurs génétiques d'une placette expérimentale d'arbres guyanais. Nous avons montré les similitudes et complémentarités entre les classifications obtenues par SBM et par les méthodes plus classiques de Classification Ascendante Hiérarchique. Ce travail a également permis de clarifier le lien entre la taxonomie botanique et la diversité moléculaire présentes dans un échantillon.
Ensuite, pour développer l'approche TT, nous considérons la loi jointe conditionnelle du SBM comme les éléments d'un tenseur. Nous l'approchons par un tenseur de format TT, développé par Oseledets & al., 2011 où chaque élément est obtenu par produit de matrices. Cette écriture mène naturellement à la simplification des calculs de marginalisation par distributivité et séparation des variables, où les marginales sont calculées par des produits matriciels. Cependant, l'approche TT appliquée au SBM débouche sur des calculs matriciels en très grande dimension conduisant à une complexité exponentielle avec le nombre d'individus. Nous avons utilisé le format TT-matrix pour les calculs matriciels, et avons traité les difficultés suivantes apparues lors des calculs : (i) éviter des nombres trop petits par une procédure d'homothétie (ii) contrôler le rang des TT-matrices par un choix des paramètres à l'entrée de l'algorithme du «rounding» proposée par Oseledets sans perdre en précision. (iii) limiter l'ordre des TT-matrice par une agrégation de ces cores.
Ainsi, les contributions de la thèse sont les suivantes : (i) une écriture exacte de la loi jointe conditionnelle d'un modèle SBM comme un tenseur au format TT, sans approximations, en tirant parti du fait que les facteurs sont au plus binaires, cette approche peut s'appliquer plus généralement à tout modèle graphique dont les facteurs sont au plus binaires ; (ii) un algorithme de type programmation dynamique pour le calcul des marginales binaires ; (iii) une procédure opérationnelle qui intègre quelques solutions aux verrous numériques présentés plus haut.
Enfin, nous avons comparé sur une grande variété de modèles SBM la précision du calcul des marginales unaires et binaires, et les temps du calcul obtenus par l'approche TT et par trois autres méthodes : simple énumération, échantillonneurs de Gibbs, approximation par champ moyen. L'approche TT est plus précise que l'approximation par champ moyen et plus rapide que l'échantillonneur de Gibbs sans perdre en précision.

Mots-clés : Modèle à bloc stochastique, calcul haute performance, algèbre linéaire.

Tensor-Train approach for inference in stochastic block models, application to biodiversity characterisation

Abstract:

The Stochastic Block Model (SBM) is a particular graphical model for clustering individuals on the basis of their pairwise distances. It builds groups such as individuals in a given group have the same pattern of connections to the other groups and to their own group. The estimation of the parameters of a SBM is classically done by the EM (Expectation-Maximisation) algorithm which at each iteration requires the computation of the unary and binary marginals of the conditional joint distribution of the groups of individuals. One way of doing this computation is to use a mean field approximation of the model where the binary marginals are computed as the product of the unary marginals. This is what is implemented in the Variational EM algorithm. This algorithm leads to good quality estimators. However, one can hope to improve the inference of the binary marginals. The objective of this thesis work is to propose a more accurate approach to the computation of binary marginals using tensor algebra, by a Tensor Train (TT) approximation, extending the approach of Novikov & al, 2014 on the computation of the partition function of a graphical model. As a first step, in order to motivate this methodological work on SBMs, we performed a clustering of genetic distance matrices in a sample of genetic markers from a Guianese tree experimental plot. We showed the similarities and complementarities between the classifications obtained by SBM and by the more classical methods of Agglomerative Hierarchical Clustering (AHC). This work allowed us to clarify the link between botanical taxonomy and the molecular diversity present in a sample. Then, to develop the TT approach, we consider the conditional joint law of the SBM as the elements of a tensor. We approximate it by the TT format tensor, developed by Oseledets & al., 2011 where each element is obtained by a product of matrices. This format naturally leads to simplification of marginalization computations by distributivity and separation of variables, where the marginals are computed by matrix products. However, the TT approach applied to SBM leads to very high dimensional matrix computations leading to an exponential complexity according to the number of individuals. We used the TT-matrix format for the matrix computations, and dealt with the following difficulties that arose during the computations: (i) avoiding too small numbers by a homothetic procedure (ii) control the rank of the TT-matrix by a choice of parameters at the input of the rounding algorithm proposed by Oseledets without losing precision (iii) to limit the order of the TT-matrix by an aggregation of these cores.

Thus, the contributions of the thesis are the following: (i) an exact writing of the conditional joint law of a SBM model as a tensor in TT format, without approximations, taking advantage of the fact that the factors are at most binary. This approach can be applied more generally to any graphical model whose factors are at most binary; (ii) a dynamic programming type algorithm for the computation of binary marginals in this context; (iii) an operational procedure that integrates some solutions to the numerical problems presented above.

Finally, we have compared the accuracy of the computation of unary and binary marginals, and the computation times obtained by the TT approach and by three other methods: simple enumeration, Gibbs samplers, mean-field approximation, on a large variety of SBM models. The TT approach is more accurate than the mean field approximation and faster than the Gibbs sampler without losing accuracy.

Keywords: Stochastic block model, high performance computing, linear algebra

---

Unité de recherche

INRAE, UMR BioGeCo, Pierroton & EPC INRAE/INRIA Pléiade, 33400 Talence, France.

Remerciements

Cette thèse est la suite d'un stage de master 2 avec comme titre "Identification d'OTU par clustering par les modèles à blocs stochastiques" encadré par Nathalie Peyrard et Alain Franc. Ce stage s'est poursuivi par le dépôt d'un projet de thèse Inrae/Inria encadré par Alain Franc, Nathalie Peyrard et Olivier Coulaud qui a été accepté.
Et après un peu plus de trois ans d'aventure, où j'ai travaillé au sein del'unité Miat de l'INRAE Toulouse (première année) et au sein de l'équipe projet commune (EPC) l'INRAE, Biogeco/INRIA, BSO Pléiade, j'ai pu finaliser ma thèse.

Dans un premier lieu, je tiens à exprimer mes sincères remerciements à mes directeurs de thèse : Olivier Coulaud et Nathalie Peyrard pour leur soutien continu et leur engagement passionné tout au long de mon parcours de recherche. Leurs conseils avisés, leurs discussions stimulantes et leur encouragement constant ont joué un rôle déterminant dans la réalisation de cette thèse.
Je souhaite aussi rendre grâce à Alain Franc, qui avait assuré l'encadrement de cette thèse du 1er octobre 2019 au 30 Avril 2021 avant de prendre sa retraite le 1er mai 2021. Je souhaite aussi le remercier pour son implication dans l'équipe d'encadrement jusqu'à la fin de ma thèse conformément à sa mission Inrae/Inria. Sa disponibilité pour discuter de mes idées, résoudre les problèmes et orienter mes recherches a été un atout précieux qui a façonné la qualité de ce travail.

Je tiens à exprimer mes sincères remerciements aux membres éminents de mon comité de thèse pour leur expertise, leur temps précieux et leurs précieuses contributions à l'évaluation de ce travail de recherche.
Je suis honoré(e) d'avoir bénéficié de la supervision, des commentaires et des conseils de Mme Sophie DONNET, M. Jean-René POIRIER, Mme Agnès BOUCHEZ et M. Pierre-Henri WUILLEMIN. Leurs réflexions perspicaces et leurs observations constructives ont grandement enrichi ma recherche, m'aidant à améliorer la pertinence de mon travail.

Je tiens ensuite à remercier mon ancien co-bureau et chef d'équipe Jean-Marc Frigerio à la fois pour les discussions que nous avons eues ainsi que l'aide qu'il m'avait apporté.

Je souhaite également exprimer ma reconnaissance envers mes collègues, qui ont enrichi mon expérience de thèse grâce à leur collaboration et leur camaraderie. Pour cela, je commencerai par remercier par collègues à la fois pour nos discussions à midi, les différentes soirées que nous avons fait ainsi que les parties de tarot et de flag-rugby.
Leur enthousiasme collectif et leur désir de réussir ont créé un environnement stimulant où l'apprentissage et l'échange d'idées étaient encouragés.

Je tiens aussi à remercier l'ensemble des équipes, des départements ainsi que les différents directeurs d'unité, pour leur soutien logistique et administratif. Leur dévouement discret, mais essentiel, a rendu mon parcours de recherche plus fluide et plus agréable.

Je tiens à exprimer ma plus profonde gratitude envers mes parents et ma sœur pour leur soutien inconditionnel tout au long de cette aventure académique. Votre amour, vos encouragements et vos sacrifices ont été les piliers qui m'ont permis d'atteindre ce moment important de ma vie.
En somme, ma réussite dans ce projet de thèse est le fruit de l'amour, du soutien et de la patience que ma famille a investie en moi. Votre présence dans ma vie a illuminé chacune de mes journées, et je suis profondément reconnaissant d'avoir une famille aussi exceptionnelle. Que ces lignes de remerciement soient le reflet de ma gratitude éternelle envers vous tous.
Je souhaite également adresser mes remerciements à ma famille élargie et à mes proches amis, dont le soutien indéfectible et les encouragements tout au long de cette aventure ont été une source de réconfort.
Je tiens à adresser mes remerciements les plus chaleureux à ma copine, Élodie Guionet, pour son soutien inébranlable et sa présence aimante qui ont été des éléments essentiels qui ont rendu ce voyage possible.

Table des matières

Table des figures

Liste des tableaux

List of Algorithms

Chapitre 1

Introduction générale

Table des matières

1.1 Contexte scientifique

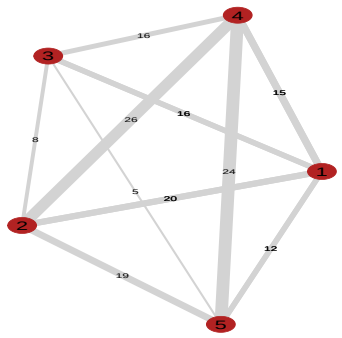
Les interactions (similarités, relations, amitiés, connaissances, collaborations, flux d'informations et communications ...) entre des entités (individus, objets, organisations, pays ...) peuvent être modélisées par des graphes mathématiques où chaque entité est représentée par un sommet et chaque interaction entre des sommets est représentée par une arête. Ces interactions peuvent être binaires ou pondérées par une distance. Dans ce cas, nous sommes dans le cadre de graphes pondérés (weighted graph [FHP91]), dans lequel un nombre (poids) est attribué à chaque arête. La représentation en graphe a vu le jour en 1736 dans un article publié par Leonhard Euler [BLW86] avant que l'appellation graphe ne soit initialement introduite en 1878 par Joseph Sylvester dans un article publié dans la revue Nature [Syl].
Presque un demi-siècle plus tard, Paul Erdős and Alfréd Rényi ont introduit des méthodes probabilistes en théorie des graphes dans leur article [ER59] paru en 1959. Depuis, avec le développement des outils de calcul et des méthodes statistiques, plusieurs chercheurs se sont intéressés aux graphes et ont développé des algorithmes et des méthodes permettant d'étendre les applications des graphes dans des domaines variés tels que l'écologie [JBO02], la biologie [Les06], l'épidémiologie [BMD20], l'économie [MB09], le traitement d'images [LMZ+20], ou les sciences sociales [TL10] ...

Dans la littérature, nous pouvons trouver plusieurs types de graphes. Nous en citons deux : les graphes orientés où les liens entre les sommets possèdent une orientation et les graphes non orientés où les liens entre les sommets ne possèdent pas d'orientation. Dans cette thèse, nous nous intéressons au cas de graphes non orientés pondérés, c'est à dire où les liens possèdent un poids. Nous présentons ci-dessous un exemple de graphe non orienté pondéré pour représenter les interactions entre cinq individus selon une matrice de distance entre les individus deux à deux.

Matrice de distances

$$\begin{pmatrix} 0 & 20 & 16 & 15 & 12 \\ 20 & 0 & 8 & 26 & 19 \\ 16 & 8 & 0 & 16 & 5 \\ 15 & 26 & 16 & 0 & 24 \\ 12 & 19 & 5 & 24 & 0 \end{pmatrix}$$

Représentation en graphe



Dans cette représentation, l'épaisseur des liens est proportionnelle à la distance entre les individus.

La popularité croissante des graphes vient de leur capacité à permettre une visualisation des interactions entre des entités de manière très concise. De plus, ils permettent de trouver la structure communautaire entre des entités, ce que nous appelons partitionnement ou clustering des sommets : il s'agit de regrouper n sommets en Q classes selon certaines propriétés d'homogénéité à partir des informations qui se trouvent dans une matrice de distances $D \in \mathbb{R}^{n \times n}$ entre individus.
Une façon de faire le partitionnement est de considérer chaque individu comme une variable aléatoire et chaque lien est dépendent de ces variables. Un modèle graphique [Bis06] décrit alors la loi de la distribution jointe entre les variables aléatoires.
C'est le cas des modèles à blocs stochastiques (Stochastic Block Model, SBM), un cas particulier

de modèles graphiques où chaque facteur n'implique que deux individus. Ces modèles ont été initialement introduits dans le domaine des réseaux sociaux en 1983 dans l'article de Holland et al [HLL83]. Ils permettent de modéliser des interactions par paires d'entités. Ces interactions sont représentées par des facteurs binaires. Ils trouvent actuellement des applications dans plusieurs autres domaines comme les sciences sociales [KN11, BDLBH15], la biologie [FYZS18] ou l'écologie [CLBD22].
Ces modèles sont plus flexibles que d'autres approches de partitionnement étant donné qu'ils construisent des classes d'individus partageant les mêmes comportements : les distances entre individus d'une même classe suivent une même loi, et les distances entre paires d'individus appartenant à une classe q et une classe q' suivent une même loi également, fonction de q et q', mais sans à priori sur les distances intra et inter classe, elles peuvent être faibles ou élevées.
Pour inférer les classes d'un SBM, il faut estimer ses paramètres : une matrice de distances entre classes et un vecteur de probabilités d'appartenance aux classes.

1.2 Intérêt et verrous dans l'inférence des marginales conditionnelles d'un SBM

Dans le cadre des SBM, le calcul des marginales unaires et binaires de la loi jointe des états des sommets du graphe (Z) sachant les distances (D) s'avère utile pour des données en écologie. Nous présentons ici deux exemples. Puis nous expliquons la complexité du calcul des marginales.

1.2.1 Intérêt en écologie

En écologie, nous pouvons être amenés à calculer des marginales unaires et binaires dans ces cas de figure.
On suppose qu'on dispose d'une communauté composée de plusieurs individus eux même regroupés en taxons où nous n'avons pas accès aux interactions (prédations, parasitisme, compétition ...) entre les différents taxons.
Une manière de quantifier la possibilité d'interactions est de travailler sur les co-occurences entre taxons. Étant donnés deux taxons i et j, les marginales binaires $p_{ij}(z_i, z_j)$ où z_i (resp. z_j) sont la présence / absence du taxon i (resp. j) représentent la loi de co-occurence des deux taxons. L'existence d'interactions peut être une hypothèse en cas d'écart par rapport à l'hypothèse neutre [Hub01] où cette marginale binaire est le produit des deux marginales unaires : $p_{ij}(z_i, z_j) \approx p_i(z_i)p_j(z_j)$. Le modèle neutre stipule que les interactions n'interviennent pas dans la diversité et la distribution des taxons, qui dépendent uniquement des processus démographiques (taux de mortalité, natalité, immigration). Nous calculons donc

$$\delta_{i,j}(z_i, z_j) = p_{ij}(z_i, z_j) - p_i(z_i)p_j(z_j)$$

S'il n'y a aucune interaction entre les deux taxons, alors

$$\delta_{i,j} \approx 0$$

S'il y a des interactions entre les individus, alors, par exemple

- si $\delta_{i,j}(1,1) > 0$, nous pouvons faire l'hypothèse d'une interaction mutualiste
- si $\delta_{i,j}(1,1) < 0$, nous pouvons faire l'hypothèse d'une interaction compétitive
- si $\delta_{i,j}(1,0) > 0$, nous pouvons faire l'hypothèse que i est prédateur ou parasite de j
- ...

Afin de faire une classification non supervisée d'individus en OTUs sur une base moléculaire avec un SBM [APF22] comme nous avons montré dans le chapitre 3. Nous sommes amenés à calculer la probabilité que deux individus soient dans le même bloc sachant la matrice de distance. En effet, l'estimation des paramètres d'un SBM se fait via l'algorithme Expectation-Maximisation (EM [DLR77]) où le calcul de ces marginales constitue l'étape expectation de chaque itération de cet algorithme comme nous l'expliquons dans la section 1.2.2 et le chapitre 2. C'est l'étape la plus coûteuse en temps vu qu'elle nécessite un nombre de sommes exponentiel par rapport au nombre d'individus si le calcul est fait de manière brutale.

1.2.2 Estimation des SBM

Une première méthode d'estimation des paramètres d'un SBM a été proposée en 1997 par Snijders et Nowicki dans l'article [Sni97]. Dans leur article, ils l'ont fait pour des SBM avec deux classes et pour des matrices D de petites tailles par l'intermédiaire d'une approximation de l'algorithme Expectation-Maximisation (EM) développé par Dempster et Al. [DLR77] en 1977. Dans l'algorithme EM, chaque itération se décompose en deux étapes : une étape de calcul d'espérance (E) et une étape de maximisation (M). Dans l'étape E, il faut calculer des marginales unaires et binaires de la loi jointe conditionnelle des individus comme nous l'expliquons dans le chapitre 2.
Cette étape est très coûteuse car pour un SBM avec n individus et Q classes, chaque marginale nécessite d'effectuer Q^n sommes, nous devons calculer de l'ordre de $\frac{n^2Q^2}{2}$ marginales binaires. Donc si n est grand ce calcul devient impossible.
J.-J. Daudin & Al. ont proposé dans l'article [DPR08] paru en en 2008 une méthode pour faire du EM approché pour l'estimation des SBM, elle consiste à procéder par une approximation champ moyen du modèle qui repose sur l'hypothèse de l'indépendance des classes des sommets sachant la matrice de distances D pour approcher la distribution de la loi jointe d'un SBM connaissant les distances par une distribution plus simple à calculer. Ainsi, les marginales binaires sont calculées comme produit des marginales unaires. Cela conduit à l'algorithme Variational EM (VEM). Cet algorithme mène à des estimateurs de bonne qualité.
D'autres méthodes probabilistes existent, comme par exemple l'approche proposée par Snijders et Nowicki dans l'article [NS01a], qui repose sur l'utilisation de l'échantillonneur de Gibbs pour calculer la loi jointe puis de l'utiliser pour calculer les marginales ou l'algorithme Bayes variationnel EM (VBEM [LBA12, LR13]).

1.2.3 Coût du calcul des marginales

Dans la section précédente, nous avons expliqué l'intérêt du calcul des marginales. Nous présentons dans cette section le coût d'un calcul de marginale pour un SBM avec n individus et Q classes. La loi jointe conditionnelle de ce modèle connaissant les distances D peut s'exprimer comme une fonction multivariée :

$$\begin{aligned} \Psi : [\![1,Q]\!]^n &\xrightarrow{\Psi} \mathbb{R} \\ (z_1,\dots,z_n) &\longrightarrow \Psi[z_1,\dots,z_n] = \prod_{i=1,n}\prod_{j=i+1,n} \psi_{i,j}(z_i,z_j) \end{aligned} \tag{1.2.1}$$

Avec $\psi_{i,j}$ le facteur associé au lien entre i et j. z_k l'état du sommet k.

Les marginales unaires de ce modèle s'expriment :

$$p_i(z_i) = \sum_{z_1,\dots,z_{i-1},z_{i+1},\dots,z_n} \Psi(z_1,z_2,\dots,z_n) \tag{1.2.2}$$

Les marginales binaires s'expriment :

$$p_i(z_i, z_j) = \sum_{z_1,\dots,z_{i-1},z_{i+1},\dots,z_{j-1},z_{j+1},\dots,z_n} \Psi(z_1, z_2, \dots, z_n) \tag{1.2.3}$$

Le calcul d'une marginales unaire nécessite Q^{n-1} sommes et celui d'une marginale binaire nécessitent Q^{n-2} sommes. Ces calculs deviennent difficiles dès lors qu'on augmente le nombre de classes ou le nombre d'individus.

1.3 Objectifs de la thèse et approches choisies

L'objectif de cette thèse est d'améliorer l'inférence des marginales binaires d'un SBM. Pour cela, nous proposons de considérer la loi jointe conditionnelle du SBM comme les éléments d'un tenseur et de réaliser l'inférence des marginales binaires par l'intermédiaire de calculs algébriques. Nous prenons le choix d'un format algébrique particulier pour exprimer ce tenseur, le format "Tenseur Train" (Tensor Train TT [Ose09, OTZ11, Ose11]) qui permet une séparation des variables z_i, donc permet le calcul des marginales. Cependant, si l'écriture algébrique est exacte, les objets (matrices, tenseurs) manipulés sont d'ordres ou de dimensions élevés, et il faudra examiner attentivement les obstacles au passage à l'échelle d'un point de vue numérique.

1.3.1 Tenseur de la loi jointe d'un SBM

La loi jointe conditionnelle d'un SBM avec n individus (m liens) et Q classes s'exprime :

$$\Psi[z_1, \dots, z_n] = \Pi_{i=1,n} \Pi_{j=i+1,n} \psi_{i,j}(z_i, z_j) \tag{1.3.1}$$

Elle peut être vue comme une fonction multivariée comme dans la formule (1.2.1). Si nous considérons les z_i comme des indices sur $[\![1, Q]\!]$ alors ψ définie un tableau multidimensionnel $\mathbf{\Psi} \in (\mathbb{R}^Q)^{\otimes n}$ tel que,

$$\mathbf{\Psi}[z_1, \dots, z_n] \in \mathbb{R} \tag{1.3.2}$$

$\mathbf{\Psi}$ est un tenseur, un objet mathématique très étudié issu de l'algèbre multilinéaire. Nous mobiliserons donc les concepts algébriques et des opérations tensorielles [Hac19, KB09, Fra22a] pour manipuler la loi jointe.

1.3.2 Format "Tenseur-Train" de la loi jointe

Les tenseurs de la loi jointe $\mathbf{\Psi} \in (\mathbb{R}^Q)^{\otimes n}$ sont de tailles exponentielles par rapport à n, ils sont donc difficilement manipulables. Si nous prenons un exemple de SBM avec un nombre d'individus $n = 40$ et de classes $Q = 3$, nous nous retrouvons avec des tenseurs de tailles 1.22×10^{19}. Oseledets & al. ont proposé un format "Tenseur Train" dans les articles [Ose09] et [OTZ11]. Il s'agit d'une approche qui permet de les approcher par des tenseurs de rang très faibles. Ce format présente de nombreux avantages. Premièrement, il permet de faire un stockage des tenseurs avec moins d'espace mémoire : nQr^2 au lieu de Q^n. Il permet de faire des opérations efficaces directement au format "Tenseur Train" (produits, sommes ...) et permet de faire la séparation des variables, ce qui facilite le calcul des marginales.
Il a été utilisé dans de nombreux articles, par exemple dans [ZYO+15] pour simuler de manière hiérarchique certains circuits/systèmes stochastiques difficiles. Dans l'article [NPOV15b], l'auteur utilise ce format pour le stockage des lawyers dans les réseaux de neurones. Dans l'article [DBNW20], les auteurs présentent une représentation pour approcher les fonctions potentiels d'un modèle graphique complexe.

Novikov & al, ont utilisé ce format dans un article [NROV14] pour le calcul de la constante de normalisation et des marginales unaires d'un modèle graphique. L'objectif de cette thèse est d'étendre ce format pour le calcul des marginales binaires d'un SBM.

1.4 Thèse pluridisciplinaire

Cette thèse s'inscrit dans un cadre pluridisciplinaire. La première partie combine le barcoding et les statistiques, elle porte sur une application des SBM pour faire une classification non supervisée sur des données moléculaires issues d'arbres de Guyane [CMS+19b]. L'objectif de ce travail est double, dans un premier temps, nous cherchons à savoir si les classifications SBM apportent des informations supplémentaires vis-à-vis des classifications CAH quand nous les comparons à la classification botanique. D'autre part, ce travail a pour objectif de clarifier le lien entre la taxonomie botanique et la diversité moléculaire présentes dans un échantillon.

La deuxième partie de cette thèse combine de l'algèbre tensorielle et les statistiques pour appliquer l'approche présentée par Novikov [NROV14] pour calculer les marginales binaires d'un SBM. Cette approche TT appliquée au SBM débouche sur des calculs matriciels en très grande dimension conduisant à une complexité exponentielle avec le nombre d'individus. Nous avons utilisé le format TT-matrix pour les calculs matriciels ce qui nous a conduit à aborder les difficultés suivantes.

1. apparition de nombres trop petits dans les calculs des marginales ou de la constante de normalisation.
2. évolution exponentielle des rangs des TT-matrices lors des calculs intermédiaires.
3. Augmentation quadratique par rapport au nombre d'individus de l'ordre des TT-matrices.

Afin de valider empiriquement la méthode, nous avons comparé les marginales unaires et binaires obtenues par l'approche TT avec celles obtenues avec la méthode de calcul exacte par énumération qui consiste à construire le tenseur de la loi jointe puis faire le calcul des marginales par sommation, à une approche de type Méthode de Monte-Carlo par chaîne de Markov [And01] (MCMC) et à la méthode variationnelle basée sur l'approximation en champ moyen [DPR08] de la loi de Z sachant D. Nous faisons ces comparaisons sur six structures de distances.

1.5 Structure du manuscrit

Ce manuscrit est structuré en sept chapitres .

Dans le chapitre 2 (Prérequis pour les modèles à blocs stochastiques), nous présentons des prérequis sur les SBM, qui seront utilisés tout au long de ce manuscrit. Nous explicitons les différents types de SBM utilisés, leurs hypothèses, leurs paramètres et comment inférer les paramètres de ces modèles.

Dans le chapitre 3 (Accord entre les classifications botaniques et moléculaires pour des niveaux taxonomiques élevés), nous présentons une synthèse de l'article [APF22] (disponible dans l'annexe C) où nous avons appliqué les SBM dans le contexte du barcoding. Nous commençons par une mise en contexte où nous expliquons le choix des SBM, le protocole expérimental et les résultats de comparaison avec la CAH et la botanique.

Le chapitre 4 (Approche "Tenseur-Train" pour la séparation des variables dans les modèles

graphiques) est structuré en deux parties. La première partie est consacrée à l'explication des formats algébriques que nous utilisons tout au long de ce manuscrit. Cette partie se subdivise en trois sections : dans la section 4.2, nous donnons quelques rappels d'algèbre linéaire. Dans la section 4.3 nous présentons le TT-format d'une loi jointe et sa représentation à l'aide de contraction dans le format Tensor Network. Dans la section 4.4, nous présentons les formats TT-matrice et TT-vecteur qui seront utilisés dans les chapitres 5 et 7.
La deuxième partie se subdivise en deux sections. La section 4.5 présente la mise au format TT d'une loi jointe. La section 4.6 présente un résumé de l'approche de Novikov [NROV14] pour le calcul des marginales d'un modèle graphique.

Dans le chapitre 5 (Approche de type Tenseur-Train pour le calcul des marginales binaires dans un modèle à blocs stochastiques), nous commençons par faire une adaptation de l'approche de Novikov aux cas des SBM dans la section 5.2. Nous présentons ensuite dans la section 5.3 un algorithme de type programmation dynamique qui met en oeuvre cette méthode de calcul des marginales binaires. Il repose sur une mutualisation des calculs basée sur une organisation des calculs sur un arbre.

Dans le chapitre 6 (Évaluation numérique des calculs des marginales d'un SBM par l'approche TT), nous présenterons quatre méthodes pour le calcul des marginales dans la section 6.2, les structures de distances simulées dans la section 6.3, le protocole pour le calcul des marginales sur les six structures dans la section 6.4, les résultats numériques dans les sous-sections 6.5 et 6.6 et enfin une discussion dans la section 6.7.

Dans le chapitre 7 (Vers le passage à l'échelle de l'approche TT), nous présentons trois verrous numériques rencontrés lors de la déclinaison de la procédure de Novikov sur les SBM et les solutions apportées. Nous commençons par expliquer la procédure d'homothétie qui permet d'éviter les problèmes des petites valeurs dans la section 7.2. Ensuite, nous présenterons une procédure de changement de paramètres dans la compression par l'algorithme du rounding dans la section 7.3. Enfin, nous expliquons une procédure de réduction des dimensions de TT-matrices de rang 1 par fusion des cores dans la section 7.4. Pour chacune des procédures, nous présentons la motivation, la mise en œuvre et des résultats numériques.

Nous clôturons ce manuscrit par le chapitre 8 qui comporte une conclusion générale dans laquelle nous présentons un résumé des chapitres de ce manuscrit et des contributions de cette thèse. Enfin, nous proposons quelques perspectives de poursuite de cette thèse.

Les chapitres 2 et 4 présentent l'état de l'art respectivement pour l'inférence des SBM et l'approche TT pour le calcul des marginales unaires d'un modèle graphique. Les chapitres 5, 6 et 7 présentent chacun une contribution de ce travail au calcul des marginales binaires d'un SBM.

Chapitre 2

Prérequis sur les modèles à blocs stochastiques

Table des matières

2.1 Introduction

Supposons que nous disposons de $n \in \mathbb{N}$ individus et d'un tableau de distances (ou de dissimilarités) $D \in \mathbb{R}^{n \times n}$ entre les individus pris deux à deux. Nous souhaitons regrouper ces individus en Q classes à partir des informations dans cette matrice. Une manière de grouper les individus dans des classes consiste à procéder par une classification non supervisée (clustering) qui consiste à attribuer la même classe à des individus proches les uns des autres et, idéalement, loin des autres classes. Ces classifications, comme la Classification Ascendante Hiérarchique (CAH [Wis69]), reposent sur le principe suivant : "*qui se ressemble s'assemble* ". Ainsi, si nous appliquons ces méthodes sur la matrice de distances D, nous aurons une partition avec des classes contenant des individus proches.
Une autre manière de regrouper les individus est d'utiliser les modèles à blocs stochastiques [HLL83, NS01b, FH82, LW19, WW87]. Basés sur un modèle statistique, ils construisent des classes d'individus partageant les mêmes comportements. Un SBM est plus flexible qu'une CAH car les individus d'une même classe ne sont pas nécessairement proches les uns des autres.
Pour résumer, les SBM permettent de regrouper les individus en fonction de leur profil de distances et non par leur proximité avec les individus appartenant à la même classe : en d'autres termes, les SBM produisent des classes et non des communautés. Nous présentons deux structures de distances que peuvent modéliser les SBM pour trois classes.

Associative [KBKK]

$$\Lambda_{\text{asso}} = \begin{pmatrix} 2 & 10 & 10 \\ 10 & 3 & 10 \\ 10 & 10 & 4 \end{pmatrix} \tag{2.1.1}$$

Dissociative [KBKK]

$$\Lambda_{\text{diss}} = \begin{pmatrix} 8 & 3 & 3 \\ 3 & 10 & 3 \\ 3 & 3 & 9 \end{pmatrix} \tag{2.1.2}$$

Les coefficients des matrices Λ_{diss} et Λ_{asso} représentent la distance moyenne entre deux individus selon leurs classes.
Nous remarquons que dans la structure de distances (2.1.1), les individus ont une faible distance (forte connexion) avec ceux qui appartiennent à leur classe et une forte distance (faible connexion) avec les individus qui n'appartiennent pas à la même classe. Il s'agit du cas où les méthodes classiques de classification non supervisées donnent de bons résultats.
Dans la structure de distances (2.1.2), les individus ont une forte distance (faible connexion) avec ceux qui appartiennent à leur classe et une faible distance (forte connexion) avec les individus qui n'appartiennent pas à la même classe. Il s'agit du cas où les méthodes classiques de classification non supervisées ne sont pas adaptées pour trouver les classes. Les SBM arrivent à trouver ces deux structures.

Ces modèles sont utilisés dans divers domaines comme la détection de communautés dans les science sociales [KN11, BLZ16, LBA11, BDLBH15], par exemple, l'article [BDLBH15] présente une application de ces modèles pour modéliser les interactions des chercheurs français sur le cancer par l'intermédiaire des connexions directes entre eux ou les connexions via leurs laboratoires de recherche. Ces modèles trouvent également une utilisation dans des domaines comme la biologie, l'article [FYZS18] présente les composants fonctionnels dans les réseaux d'interaction protéine-protéine. Ils peuvent aussi être utilisés en écologie : l'article [CLBD22] présente une utilisation dans la modélisation des réseaux d'interactions écologiques bipartites. Un autre cas d'usage pour les SBM est la génération de topologies Internet et IP synthétiques [KBKK].
Dans la littérature, nous pouvons distinguer deux types de SBM :

- les SBM binaires [DPR08, BDLBH15, KBCG12] (binary SBM) où les tableaux distances représentent la présence/absence de liens entre deux individus. Cette distance est la réalisation d'une loi de Bernoulli ;

- les SBM pondérés par les distances [JZS09, MRV10] (weighted SBM).

Nous nous intéressons aux SBM pondérés par les distances. Nous expliquons leurs hypothèses et nous décrivons leurs paramètres, leurs lois jointes, l'estimation des paramètres de ces modèles par l'intermédiaire de l'algorithme Expectation-Maximisation (EM [DLR77]) ainsi que les difficultés de cet algorithme.

2.2 SBM pondérés par les distances

2.2.1 Hypothèses, paramètres et loi jointe des SBM

Nous disposons d'un nombre $n \in \mathbb{N}$ individus et d'une matrice $D \in \mathbb{R}^{n \times n}$ de distances entre individus deux à deux. Nous souhaitons regrouper les individus en $Q \in [\![1, n]\!]$ classes à partir des informations contenues dans la matrice D.
L'appartenance de chaque individu i à chaque classe q est représentée par une variable latente $Z = (Z_1, ..., Z_n)$ avec $\forall\, i = 1, ...n$, $Z_i \in [\![1, Q]\!]$. Cela se note également selon une matrice $n \times Q$ avec :

$$\begin{cases} Z_{i,q} = 1 & \text{Si,} \quad Z_i = q \\ Z_{i,q} = 0 & \text{Sinon} \end{cases} \tag{2.2.1}$$

($z_{i,q} = 1$ si l'élément i est dans la classe q et 0 sinon).
Par construction,

$$\forall\, i, \quad \sum_{q=1}^{Q} z_{i,q} = 1 \tag{2.2.2}$$

Nous nous intéresserons dans la suite de cette section aux SBM pondérés avec des distances suivant une loi de Poisson et une loi gaussienne.

2.2.1.1 Hypothèses des modèles

Les SBM repose sur deux hypothèses :

- $\boldsymbol{H}_1$ les distances sont indépendantes sachant Z et ne dépendent que des classes des individus. La distance entre deux individus est tirée d'une distribution dépendant des classes de ces deux individus, c'est-à-dire :

$$D_{i,j} | Z_i = q, Z_j = q' \quad \sim \quad \text{Distribution}(\xi_{q,q'}) \tag{2.2.3}$$

  où la distribution est soit Gaussienne soit Poisson et $\xi_{q,q'}$ est un paramètre. Nous notons $\xi = \{\xi_{q,q'}\}_{q,q'}$.
- $\boldsymbol{H}_2$ les variables latentes Z_i sont indépendantes et identiquement distribuées (iid) selon une distribution $\alpha = (\alpha_1, \ldots, \alpha_Q)$ avec

$$\forall q \in [1, ..., Q], \qquad \alpha_q = \mathbb{P}_\alpha(Z_{i,q} = 1) \tag{2.2.4}$$

α représente les probabilités d'appartenance aux classes.

2.2.1.2 Loi jointe des SBM

Nous posons $\theta = (\xi, \alpha)$ les paramètres du modèle et nous appliquons le théorème Bayes. La loi jointe de Z et D s'exprime :

$$\mathbb{P}_\theta(Z, D) = \mathbb{P}_\xi(D \mid Z)\mathbb{P}_\alpha(Z) \tag{2.2.5}$$

Nous utilisons la première hypothèse ($\boldsymbol{H}_1$) pour exprimer la loi jointe comme :

$$\mathbb{P}_\theta(Z,D) = \prod_{i=1}^{n}\prod_{i>j}\mathbb{P}_\xi(D_{i,j}|Z)P_\alpha(Z) \tag{2.2.6}$$

$$\Rightarrow$$

$$\mathbb{P}_\theta(Z,D) = \prod_{i=1}^{n}\prod_{i>j}\left(\sum_{q=1}^{Q}\sum_{q'=1}^{Q}\mathbb{P}_\xi(D_{i,j}|z_{i,q}=1,z_{j,q'}=1)\mathbb{1}(z_{i,q}=1,z_{j,q'}=1)\right)\mathbb{P}_\alpha(z) \tag{2.2.7}$$

Nous constatons que $\mathbb{1}(z_{i,q}=1,z_{j,q'}=1)$ vaut 1 si les individus i et j appartiennent aux classes q et q'. Donc

$$\mathbb{1}(Z_{i,q}=1,Z_{j,q'}=1) = z_{i,q}\times z_{j,q'} \tag{2.2.8}$$

Nous injectons la formule (2.2.8) dans la formule (2.2.7) et nous obtenons :

$$\mathbb{P}_\theta(Z,D) = \prod_{i=1}^{n}\prod_{i>j}\left(\sum_{q=1}^{Q}\sum_{q'=1}^{Q}\mathbb{P}_\xi(D_{i,j}\mid Z_{i,q}=1,Z_{j,q'}=1)z_{i,q}\times z_{j,q'}\right)\mathbb{P}_\alpha(z) \tag{2.2.9}$$

L'hypothèse ($\boldsymbol{H}_2$) et la formule (2.2.4) nous permet d'écrire $\mathbb{P}_\alpha(z)$ de la sorte :

$$\mathbb{P}_\alpha(z) = \prod_{i=1}^{n}\prod_{q=1}^{Q}\mathbb{P}(Z_{i,q}=1)^{Z_{i,q}} = \prod_{q=1}^{Q}\alpha_q^{z_{i,q}} \tag{2.2.10}$$

La loi jointe s'exprime :

$$\mathbb{P}_\theta(Z,D) = \prod_{i=1}^{n}\prod_{i>j}\left(\sum_{q=1}^{Q}\sum_{q'=1}^{Q}\mathbb{P}_\xi(D_{i,j}\mid Z_{i,q}=1,Z_{j,q'}=1)z_{i,q}\times z_{j,q'}\right)\prod_{q=1}^{Q}\alpha_q^{z_{i,q}} \tag{2.2.11}$$

Facteurs binaires et unaires : nous notons $\phi_i(z_i)$ les facteurs unaires et $\psi_{i,j}(z_i,z_j)$ les facteurs binaires de la loi jointe :

$$\phi_i(z_i) = \prod_{q=1}^{Q}\alpha_q^{z_{i,q}} \tag{2.2.12}$$

$$\psi_{i,j}(z_i,z_j) = \sum_{q=1}^{Q}\sum_{q'=1}^{Q}\mathbb{P}_\xi(D_{i,j}\mid Z_{i,q}=1,Z_{j,q'}=1)z_{i,q}\times z_{j,q'} \tag{2.2.13}$$

La loi jointe des SBM peut s'exprimer :

$$\mathbb{P}_\theta(Z,D) = \left[\prod_{i=1}^{n}\prod_{j>i}\psi_{i,j}(z_i,z_j)\right]\left[\prod_{i=1}^{n}\phi_i(z_i)\right] \tag{2.2.14}$$

2.2.1.3 Deux exemples de paramétrisation d'un SBM pondéré

Les paramètres sont les proportions des individus dans les classes α et le paramètre ξ. Ce dernier dépend de la loi par laquelle nous modélisons les distances.

Matrice de connectivité des SBM pondérés pour des distances suivant une loi gaussienne : la probabilité conditionnelle de la distance entre deux individus sachant leurs classes suit une loi gaussienne :

$$D_{i,j}|Z_i=q,Z_j=q'\sim\mathcal{N}(\mu_{q,q'},\sigma^2)$$

Dans ce cas, $\xi=\{\mu_{q,l}\}$. Nous notons la matrice de connectivité $M\in\mathbb{R}^{Q\times Q}$ et ses coefficients $\mu_{q,q'}$.

Matrice de connectivité des SBM pondérés pour des distances suivant une loi de Poisson :

$$D_{i,j}|Z_i = q, Z_j = l \sim Poisson(\lambda_{q,q'}) \tag{2.2.15}$$

Dans ce cas, $\xi = \{\lambda_{q,l}\}$. Nous notons la matrice de connectivité $\Lambda \in \mathbb{R}^{Q\times Q}$ et ses coefficients $\lambda_{q,q'}$.

2.2.1.4 Loi jointe des SBM Poisson

Nous appliquons (2.2.11) dans le cas Poisson, car c'est sur ce modèle que nous avons travaillé dans cette thèse. Nous posons $\theta = (\Lambda, \alpha)$ les paramètres du modèle. Comme la distance entre deux individus suit une loi de Poisson, alors $\mathbb{P}_\Lambda$ s'exprime :

$$\mathbb{P}_\Lambda(D_{i,j}|z_{i,q} = 1, z_{j,q'} = 1) = \frac{\lambda_{q,q'}^{D_{i,j}}}{D_{i,j}!}\exp(-\lambda_{q,q'}) \tag{2.2.16}$$

Enfin, à partir de (2.2.16) et (2.2.10), la loi jointe d'un modèle SBM Poisson s'exprime :

$$\mathbb{P}_\theta(Z, D) = \prod_{i=1}^{n}\prod_{i>j}\left(\sum_{q=1}^{Q}\sum_{q'=1}^{Q}\frac{\lambda_{q,q'}^{D_{i,j}}}{D_{i,j}!}\exp(-\lambda_{q,q'})z_{i,q}\times z_{j,q'}\right)\prod_{i=1}^{n}\prod_{q=1}^{Q}\alpha_q^{z_{i,q}} \tag{2.2.17}$$

Facteurs binaires et unaires Les facteurs s'expriment dans le cadre d'un SBM Poisson :

$$\phi_i(z_i) = \prod_{q=1}^{Q}\alpha_q^{z_{i,q}} \tag{2.2.18}$$

$$\psi_{i,j}(z_i, z_j) = \sum_{q=1}^{Q}\sum_{q'=1}^{Q}\frac{\lambda_{q,q'}^{D(i,j)}}{D(i,j)!}\exp(-\lambda_{q,q'})z_{i,q}\times z_{j,q'} \tag{2.2.19}$$

2.3 Algorithme EM pour l'estimation des paramètres d'un modèle SBM

L'estimateur du maximum de vraisemblance du modèle $\hat{\theta}_{MV}$ est défini par :

$$\hat{\theta}_{MV} = \text{argmax}(\mathbb{P}_\theta(D)) \tag{2.3.1}$$

Pour l'obtenir, il faut calculer la vraisemblance du modèle :

$$\mathbb{L}(D, \theta) = \mathbb{P}_\theta(D) = \sum_z \mathbb{P}_\theta(D, Z = z) \tag{2.3.2}$$

où $Z = (Z_1, \dots Z_n)$.
Soit dans le cas d'un modèle SBM Poisson :

$$\mathbb{L}(D, \theta) = \sum_z\prod_{i=1}^{n}\prod_{i>j}(\sum_{q=1}^{Q}\sum_{q'=1}^{Q}\frac{\lambda_{q,q'}^{D_{i,j}}}{D_{i,j}!}\exp(-\lambda_{q,q'})z_{i,q}\times z_{j,q'}))\prod_{i=1}^{n}\prod_{q=1}^{Q}\alpha^{z_{i,q}} \tag{2.3.3}$$

Une manière de trouver $\hat{\theta}_{MV}$ est de trouver les paramètres qui annulent la dérivée de la vraisemblance (2.3.3). Dériver cette quantité n'est pas très difficile, mais trouver toutes les valeurs qui l'annulent est compliqué, en particulier parce qu'il y a Q^n sommes.

Un algorithme classique qui permet de trouver le maximum de la vraisemblance dans le cas des modèles à variables latentes est l'algorithme espérance-maximisation (expectation-maximisation EM [DLR77]) qui procède de manière itérative.
Comme son nom l'indique, chaque itération de cet algorithme se décompose en deux étapes : une étape de calcul d'espérance (E-step) et une étape de maximisation (M-step).
L'algorithme EM consiste à :

1. **Initialisation** : choisir une valeur initiale pour les paramètres : $(\alpha^0, \lambda^0) = \theta^0$.
2. **Alternance des étapes E et M** jusqu'à ce que le critère d'arrêt soit vérifié. Supposons que la valeur courante soit θ^t, à l'itération $t+1$:
   — **Étape E** : Calcul de toutes les marginales binaires conditionnelles (les $\mathbb{P}_{\theta^t}(Z_i, Z_j \mid D)$ pour tous les couples Z_i, Z_j avec $i < j$) et unaires (les $\mathbb{P}_{\theta^t}(Z_i \mid D)$ pour tout i) de $\mathbb{P}_{\theta^t}(Z \mid D)$ pour la valeur courante des paramètres.
   — **Étape M** : mise à jour des paramètres, i.e. calcul de θ^{t+1},

$$\theta^{t+1} = \arg\max_{\theta} \mathbb{Q}(\theta, \theta^t) \tag{2.3.4}$$

   avec,

$$\mathbb{Q}(\theta, \theta^t) = \sum_{i=1}^{n} \sum_{j>i}^{n} \sum_{z_i=1}^{Q} \sum_{z_j=1}^{Q} \log(\mathbb{P}_{\theta^t}(D_{i,j} \mid z_i, z_j)) \mathbb{P}_{\theta^t}(z_i, z_j \mid D) + \sum_{i=1}^{n} \sum_{b=1}^{Q} \sum_{z_{i,b}} \mathbb{P}_{\theta^t}(z_{i,b} | D) z_{i,b} \log(\alpha_b) \tag{2.3.5}$$

   Cela donne les formules de mises à jours suivantes :

$$\alpha_b^{t+1} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{P}_{\theta^t}(Z_{i,b} = 1 \mid D) \tag{2.3.6}$$

$$\lambda_{b,b'}^{t+1} = \frac{\sum_{i=1}^{n} \sum_{j \neq i} D(i,j) \mathbb{P}_{\theta^t}(Z_i = b, Z_j = b' \mid D)}{\sum_{i=1}^{n} \sum_{j \neq i} \mathbb{P}_{\theta^t}(Z_i = b, Z_j = b' \mid D)} \tag{2.3.7}$$

3. **Un critère d'arrêt possible** est lorsque la somme des valeurs absolues des écarts relatifs entre les paramètres est plus petite que ϵ :

$$\sum_{q=1}^{Q} \frac{|\alpha_q^t - \alpha_q^{t+1}|}{\alpha_q^{t+1}} + \sum_{q=1}^{Q} \sum_{q'=q}^{Q} \frac{|\lambda_{q,q'}^t - \lambda_{q,q'}^{t+1}|}{\lambda_{q,q'}^{t+1}} < \epsilon$$

Á la fin de cet algorithme, nous obtenons $\hat{\theta}$ l'estimateur de θ qui est un maximum local de la vraisemblance.

2.3.1 E-step

Nous détaillons la E-step afin de montrer pourquoi dans le cadre des SBM, cela revient à calculer les marginales unaires et binaires conditionnellement à D. Pour cela, nous commençons par définir la quantité suivante $\mathbb{Q}(\theta, \theta^t)$:

$$\mathbb{Q}(\theta, \theta^t) = \mathbb{E}_Z[\log \mathbb{P}_\theta(D, Z) | \theta^t, D] \tag{2.3.8}$$

À partir de la propriété de l'espérance $\mathbb{E}(\phi(X)) = \sum_{x_i} \phi(x_i) \mathbb{P}(x_i)$, nous avons :

$$\mathbb{Q}(\theta, \theta^t) = \sum_{z} \mathbb{P}_{\theta^t}(z \mid D) \log(\mathbb{P}_\theta(D, z)) \tag{2.3.9}$$

À partir du théorème de Bayes

$$\mathbb{Q}(\theta, \theta^t) = \sum_z \mathbb{P}_{\theta^t}(z \mid D) \log(\mathbb{P}_\theta(D|z)\mathbb{P}_\theta(z)) \tag{2.3.10}$$

Ensuite, à partir de la propriété du logarithme

$$\mathbb{Q}(\theta, \theta^t) = \sum_z \log(\mathbb{P}_\theta(D|z))\mathbb{P}_{\theta^t}(z \mid D) + \sum_z \log(\mathbb{P}_\theta(z))\mathbb{P}_{\theta^t}(z \mid D) \tag{2.3.11}$$

Nous remarquons que $\mathbb{P}_\theta(z)$ ne dépend que de α et que $\mathbb{P}_\theta(D|z)$ ne dépend que de Λ. Nous pouvons donc définir les deux quantités suivantes :

$$\mathbb{Q}(\Lambda, \theta^t) = \sum_z \log(\mathbb{P}_\theta(D|z))\mathbb{P}_{\theta^t}(z \mid D) \tag{2.3.12}$$

$$\mathbb{Q}(\alpha, \theta^t) = \sum_z \log(\mathbb{P}_\alpha(z))\mathbb{P}_{\theta^t}(z \mid D) \tag{2.3.13}$$

Nous détaillons l'expression de chacune :

Calcul de $\mathbb{Q}(\Lambda, \theta^t)$: Nous avons

$$\mathbb{Q}(\Lambda, \theta^t) = \sum_z \mathbb{P}_{\Lambda^t}(z \mid D) \log \prod_{i=1}^{n} \prod_{j>i}^{n} \mathbb{P}_\Lambda(D_{i,j}|z_i, z_j) \tag{2.3.14}$$

$$= \sum_z \mathbb{P}_{\theta^t}(z \mid D) \sum_{i=1}^{n} \sum_{j>i}^{n} \log(\mathbb{P}_\Lambda(D_{i,j}|z_i, z_j)) \tag{2.3.15}$$

Nous remarquons que $\sum_z$ peut être développée de cette manière :
$\forall i, j = 1, ..., n, \quad \sum_z = \sum_{z_1} \sum_{z_2} \sum_{z_3} \dots \sum_{z_n}$

Alors,

$$\mathbb{Q}(\Lambda, \theta^t) = \sum_{i=1}^{n} \sum_{j>i}^{n} \sum_{z_i} \sum_{z_j} \log(\mathbb{P}_{\theta^t}(D_{i,j} \mid z_i, z_j))\mathbb{P}_{\theta^t}(z_i, z_j \mid D) \tag{2.3.16}$$

La probabilité $\mathbb{P}^t_\theta(z_i, z_j \mid D)$ représente la marginale binaire conditionnellement à D. Dans la suite, nous les noterons $p_{ij}(z_i, z_j)$.

Calcul de $\mathbb{Q}(\alpha, \theta^t)$: Nous avons,

$$\mathbb{Q}(\alpha, \theta^t) = \sum_z \log \mathbb{P}_\alpha(z) \mathbb{P}_{\theta^t}(z \mid D) \tag{2.3.17}$$

$$= \sum_z \mathbb{P}_{\theta^t}(z \mid D) \log \prod_{i=1}^{n} \prod_{b=1}^{Q} \alpha_b^{z_{i,b}} \tag{2.3.18}$$

Nous utilisons la propriété du logarithme :

$$\mathbb{Q}(\alpha, \theta^t) = \sum_{i=1}^{n} \sum_{b=1}^{Q} \sum_z \mathbb{P}_{\theta^t}(z \mid D) \log \alpha_b^{z_{i,b}} \tag{2.3.19}$$

Ensuite,

$$\mathbb{Q}(\alpha, \theta^t) = \sum_{i=1}^{n} \sum_{b=1}^{Q} \mathbb{E}_z[(\log(\alpha_b^{z_{i,b}}))|\theta_t, D] \tag{2.3.20}$$

Nous constatons que dans la formule (2.3.20), la quantité $(\log(\alpha_b^{z_{i,b}}))$ ne dépend que de $z_{i,b}$ alors, nous pouvons redévelopper la formule (2.3.20)

$$\mathbb{Q}(\alpha,\theta^t) = \sum_{i=1}^{n}\sum_{b=1}^{Q}\mathbb{E}_{z_{i,b}}[(\log(\alpha_b^{z_{i,b}}))|\theta_t, D] \tag{2.3.21}$$

Ainsi,

$$\mathbb{Q}(\alpha,\theta^t) = \sum_{i=1}^{n}\sum_{b=1}^{Q}\sum_{z_{i,b}}\mathbb{P}_{\theta^t}(z_{i,b}|D)z_{i,b}\log(\alpha_b) \tag{2.3.22}$$

La probabilité $\mathbb{P}_{\theta^t}(z_{i,b}|D)$ représente la marginale unaire. Dans la suite, nous les noterons $p_i(z_i)$.

Calcul de $\mathbb{Q}(\theta,\theta^t)$:

$$\mathbb{Q}(\theta,\theta^t) = \mathbb{Q}(\lambda,\theta^t) + \mathbb{Q}(\alpha,\theta^t) \tag{2.3.23}$$

$$\mathbb{Q}(\theta,\theta^t) = \sum_{i=1}^{n}\sum_{j>i}^{n}\sum_{z_i}\sum_{z_j}\log(\mathbb{P}_{\theta^t}(D_{i,j} \mid z_i,z_j))\mathbb{P}_{\theta^t}(z_i,z_j \mid D) + \sum_{i=1}^{n}\sum_{b=1}^{Q}\sum_{z_{i,b}}\mathbb{P}_{\theta^t}(z_{i,b}|D)z_{i,b}\log(\alpha_b) \tag{2.3.24}$$

L'étape de maximisation consiste à trouver $\hat{\theta}$ les paramètres qui maximisent (2.3.24) .

2.3.2 Difficulté de l'algorithme EM et alternative

L'algorithme Expectation-Maximisation (EM) est un algorithme très précis pour l'estimation des paramètres d'une SBM. Cependant, l'étape Expectation est très coûteuse : En effet, le calcul d'une marginale unaire nécessite Q^{n-1} sommes et celui d'une marginale binaire nécessite Q^{n-2} sommes.
De plus, il faudrait calculer toutes les marginales unaires $p_i(z_i)$ et binaires $p_{ij}(z_i,z_j)$. Dans un SBM, il y a m paires (i,j) possibles :

$$m = \sum_{i=1}^{n}\sum_{j=i+1}^{n} 1 = \frac{n(n-1)}{2} \tag{2.3.25}$$

Comme m est quadratique par rapport à n, nous aurons besoin de calculer $\mathcal{O}(n^2)$ marginales binaires à chaque itération.
Il existe plusieurs alternatives à l'algorithme EM, parmi ces alternatives, nous pouvons citer l'algorithme Variationnel EM (VEM [DPR08]) qui est basé sur l'approximation en champ moyen de la loi de Z sachant D : c'est une méthode basée sur une hypothèse d'indépendance où les marginales binaires sont obtenues comme le produit des unaires.
Dans cet algorithme, l'étape E consiste à résoudre une équation du point fixe pour obtenir les marginales unaires, puis à faire le produit des marginales unaires pour obtenir les marginales binaires. Nous détaillons cette étape dans la section 6.2.4.

2.4 Conclusion

Dans ce chapitre, nous avons présenté les SBM pondérés par des distances, nous avons expliqué leurs hypothèses, leurs paramètres, leurs lois jointes ainsi que l'estimation des paramètres de ces modèles par l'intermédiaire de l'algorithme Expectation-Maximisation. Nous avons ensuite

présenté la limite de cet algorithme qui vient du fait que nous devons calculer un très grand nombre de marginales durant l'étape E en plus de la difficulté de calculer une marginale.
Dans ce qui va suivre, nous présentons dans le chapitre 3 et l'article [APF22] une comparaison de classifications par SBM et différentes classifications ascendantes hiérarchiques (avec trois critères) sur des tableaux de distances moléculaires entre 1500 arbres de Guyane (voir [CMS+19b] pour les données).
Ensuite, nous résumons dans le chapitre 4 une approche algébrique basée sur la procédure de Novikov [NROV14] qui repose sur des tenseurs pour calculer les marginales par séparation des variables. Nous étendons cette approche aux SBM dans le chapitre 5. Dans le chapitre 6 nous comparons les résultats de cette approche avec les méthodes classiques de calcul des marginales. Enfin, dans le chapitre 7, nous proposons différentes procédures pour faire le passage à l'échelle de cette approche.

Chapitre 3

Accord entre les classifications botaniques et moléculaires pour des niveaux taxonomiques élevés

Table des matières

Dans ce chapitre, nous présentons une partie de l'article "Does clustering of dna barcodes agree with botanical classification directly at high taxonomic levels ? trees in french guiana as a case study" [APF22] que nous mettons dans les annexes.

3.1 Introduction

L'un des problèmes majeurs qui se posent dans le cadre de la classification basée sur la morphologie est que la majeure partie de la diversité reste encore inconnue. Dans l'article [LK15], l'auteur explique qu'entre 33% et 91% de toute la biodiversité marine est encore inconnue. En revanche cette classification est connue en botanique, cela nous a conduit à utiliser la botanique comme référence.
L'identification des unité taxonomique opérationnelle [1] (Operational Taxonomic Unit, OTU) dans un échantillon environnemental (ensemble de séquences) et l'organisation de la diversité moléculaire en tant que fréquence d'OTUs est une alternative à la classification botanique pour caractériser la biodiversité. Pour faire cette caractérisation, il existe deux familles d'approches :

- **Apprentissage supervisé :** Une approche classique qui consiste à construire des OTUs et à les mapper sur des bases de données de référence qui contiennent des barcodes de référence, comme fait dans l'outil BLAST [AGM+90] (Basic Local Alignment Search Tool). Cette approche est très efficace lorsque les bases de référence existent. Cependant, dès lors que ces bases sont inexistantes où incomplètes, une telle comparaison devient impossible. De plus, la plupart des classifications supervisées sont réalisées à un grain souvent beaucoup plus grossier (ordre/familles) que le niveau du genre/espèce.

- **Apprentissage non supervisé :** Cela consiste à l'attribution des taxons (aux individus en regroupant les individus dont les séquences sont proches dans une même classe par l'intermédiaire de dendrogrammes ou d'arbres.

Notre objectif dans ce chapitre et l'article [APF22] est d'étudier si la classification moléculaire par apprentissage non supervisé sur des matrices de distance en séquences deux à deux et la classification botanique convergent vers les mêmes partitions des individus, c'est à dire qu'une classe correspond à un taxon. Et, si oui, avec quelle précision.
Les méthodes classiques pour faire un apprentissage non supervisé sur des distances sont les méthodes de Classification Ascendante Hiérarchique CAH qui crée des classes d'individus similaires. Un premier objectif de notre étude serait d'utiliser la CAH avec trois critères d'agrégation : Single Linkage, Complete Linkage et Ward [Mur83] [Mü13].
Nous souhaitons aussi savoir si nous pouvons regrouper les séquences ADN avec une approche plus flexible qui consistera à regrouper dans la même classes les séquences qui ont le même profil de disctances. Pour cela, nous utiliserons les SBM.
Dans cette étude nous abordons les questions suivantes :

- Est-ce qu'il y a un accord entre la classification moléculaire et la classification botanique au niveau des espèces, genres, des familles et des ordres ?

- Est-ce que les SBM donnent les mêmes résultats que la CAH ? Ou bien apportent-ils une autre information ?

1. définition qui consiste à regrouper les individus par proximité génétiquement.

3.2 Matériel et méthode

3.2.1 Jeu de données

Nous avons mené cette étude sur un jeu de données constitué d'arbres de la parcelle expérimentale située dans la "Piste de Saint-Elie" en Guyane rassemblées pour l'étude publiée dans [CMS+19a]. Parmi ces données, nous avons sélectionné 1458 arbres. Pour chacun d'entre eux, nous disposons du nom botanique tel que donné par les botanistes de terrain et une séquence du marqueur chloroplastique trnH-psbA.
Ce jeu de données est composé de trois fichiers (voir [FCS+21] pour plus de détails) :

- un fichier csv des noms botaniques de chaque arbre pour l'ordre, la famille, le genre et l'espèce ;
- un fichier de séquences pour chaque arbre ;
- un fichier csv de distances par paires : Il a été construit par l'algorithme de Smith-Waterman pour l'alignement local des séquences [SW81]. Cette méthode est la plus précise des mesures de distances génétiques entre les séquences, mais son calcul nécessite beaucoup de temps ;
- un fichier csv des distances par paires basées sur la comparaison des histogrammes des tétramères (même format que les distances de Smith-Waterman) : Il a été construit à partir de la distance entre distributions de kmers. Nous avons choisi des kmer courts avec $k = 4$ (il y a $4^4 = 64$ tétramères différents : GTAG, TAGA, AGAG, GAGC, AGCT, GCTG, CTGT ...) ;

Nous travaillons donc sur les fichiers de distances qui représentent des matrices $D \in \mathbb{R}^{1458\times1458}$, chaque élément est la distance entre séquences.

3.2.2 Les deux types d'expériences

Dans notre étude, nous comparons les classifications botaniques et moléculaires sur deux ensembles de données de natures différentes. Nous expliquons ci-dessous la procédure de sélection des échantillons qui est la même que ce soit pour la distance kmer ou Smith-Waterman.

Comparaison sur le jeu de données complet : Le but est de faire une classification pour trouver les taxons d'un niveau de donné dans le jeu de donnée. Nous nous intéressons aux quatre niveaux taxonomiques : ordres, familles, genres et espèces afin de voir si l'accord entre les deux classifications dépend du niveau taxonomique et nous faisons notre choix de séquences de cette manière :

1. nous choisissons le niveau taxonomique.
2. nous sélectionnons les taxons de ce niveau pour lesquels nous avons un nombre raisonnable de séquences : nous avons fixé cette taille à 15 pour les ordres, 10 pour les familles et les genres et 5 pour les espèces.
3. nous construisons la sous-matrice de distance D à partir des séquences appartenant aux taxons sélectionnés.

Nous construirons donc quatre matrices de distances, une par niveau taxonomique. Si nous choisissons par exemple les familles, nous aurons une matrice de distance $D \in \mathbb{R}^{1349\times1349}$, un nombre de classes $Q = 30$ et pour chaque taxon un nombre minimal d'individus égal à 10.
Dans le tableau 3.1, nous présentons pour chaque niveau taxonomique, le nombre de séquences (taille de la matrice) dans les échantillons, le nombre de taxons et les tailles minimales des

Niveau taxonomique	Nombre de séquences	Nombre de taxons	Taille minimale
Espèces	313	55	5
Genres	845	36	10
Familles	1349	30	10
Ordres	1357	11	15

TABLE 3.1 – Caractéristiques des sous échantillons

taxons qui constituent les échantillons.

Comparaison sur 30 réplicats : Le but de ces expériences est de capter la variabilité dans les résultats de comparaison en travaillant sur les réplicats. Pour cela, nous essayons avec les SBM et la CAH de chercher les familles dans les ordres et les genres dans les familles

1. Nous choisissons le niveau taxonomique (ordre ou famille).
2. Nous sélectionnons les taxons du niveau choisi selon deux critères : Un effectif suffisant, c'est-à-dire au moins dix individus et un nombre de descendants (familles dans les ordres et genres dans les familles) qui permettrait de faire une classification (supérieur où égal à deux). Nous présentons un exemple de taxon choisi pour le niveau taxonomique ordre. Dans le jeu de données sont présentes trois familles de l'ordre des Ericales.

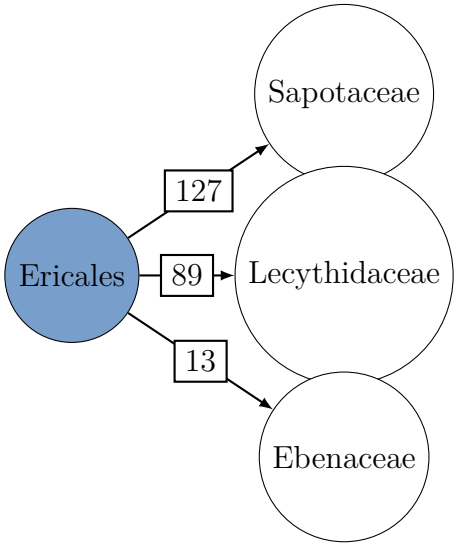


FIGURE 3.1 – Exemple de taxon choisi pour les réplicats

Dans cet exemple, l'échantillon a un effectif égal à 229 et un nombre de descendants égale à 3. Ainsi, nous construirons une matrice de distances $D \in \mathbb{R}^{229\times229}$ et les classifications SBM et CAH seront construite pour un nombre de classes $Q = 3$.

Á la fin de cette phase, nous obtenons 30 sous-échantillons : 10 ordres avec des effectifs variant entre 322 et 13 individus puis 20 familles dont les effectifs varient entre 127 et 15 individus.

3.2.3 Protocole

Afin de comparer la classification moléculaire et la classification botanique puis les classifications moléculaires entre elles, nous avons adopté le même protocole pour les deux distances (kmers et Smith-Waterman) et pour les deux types d'expériences (sur le jeu de données complet et sur les 30 réplicats). Ce protocole se décline en quatre étapes :

Étape 1 : Sélection des sous-échantillons Nous sélectionnons les sous échantillons comme présenté dans 3.2.2.

Étape 2 : Construction des partitions Pour chaque sous-échantillon et chaque matrice de distance, nous construisons des partitions des séquences à la fois avec la CAH selon trois critères d'agrégation, Ward, Complete Linkage (CL) et Single Linkage (SL) (que nous présentons dans le supplementary information (SI) de l'article [APF22]) et avec SBM.
Le nombre de clusters à trouver est Q est fourni par la classification botanique des individus du sous-échantillon. À la fin de l'étape 2, nous obtenons cinq partitions différentes pour chaque sous-échantillon.

Étape 3 : comparaison des partitions nous comparons les partitions, deux par deux, pour chaque paire possible de classifications (10 paires au total). Nous utilisons un outil visuel pour l'analyse de l'accord entre deux classifications, avec les diagrammes de Sankey [2].
Nous avons également calculé un indice pour quantifier l'accord entre les classifications, pour cela, nous avons choisi l'information mutuelle normalisée (Normalized Mutual Information NMI [PLP09]) que nous rappelons dans le supplementary information de l'article. Nous avons choisi cet indice car il repose une base solide dans la théorie de l'information, contrairement aux indices basés sur le comptage de paires d'éléments qui peuvent être non symétriques ou non bornés ou même dépendre de Q ou n, rendant la comparaison difficile. Le NMI est normalisé et varie entre 0 et 1, ce qui facilite l'interprétation et la comparaison des indices. Un NMI de 0 indique l'indépendance entre les deux classifications, tandis qu'un NMI de 1 indique un accord total.

Étape 4 : analyse des indices Nous analysons les distributions des indices obtenus pour les dix paires de classifications afin d'étudier les tendances de l'accord. Nous utilisons d'abord des représentations : histogramme, pairplots et boxplots. Ensuite, nous calculons les statistiques sur la distribution des indices (moyenne, médiane ...) que nous présentons dans les tableaux 3.3 et 3.2.

Mise en oeuvre pratique : Pour la classification SBM nous avons utilisé le package R blockmodels [Leg16] développé par Jean-Benoist & al. Leger qui permet de faire l'inférence des paramètres d'un modèle SBM par l'intermédiaire de l'algorithme VEM basé sur une approximation champs moyen.
Nous avons utilisé le package fastcluter [Mü13] disponible en R et Python pour réaliser les classifications ascendantes hiérarchiques.

3.3 Résultats

Dans cette section, nous présentons deux ensembles de résultats. Les premiers résultats portent sur les 30 sous-échantillons. Nous présentons les résultats de comparaison entre les classifications moléculaires et la classification botanique. Ensuite, nous présentons des résultats de comparaison entre les trois classifications de type CAH et la classification SBM.
Les seconds résultats sont sur le jeu de donnée complet. Pour cela, nous commençons par présenter un tracé des indices NMI selon le niveau taxonomique puis les diagrammes de Sankey.

2. Un diagramme de Sankey est un diagramme dans lequel la largeur d'une flèche est proportionnelle au flux. Par exemple, s'il y a $n_{kk'}$ séquences qui sont dans la classe k pour la classification botanique et k' pour une classification numérique, il y a un flux d'intensité $n_{kk'}$ entre ces deux clusters.

Nous présentons seulement certains résultats, l'ensemble des résultats sont présentés dans l'article [APF22] disponible dans l'annexe C.

3.3.1 Résultats des comparaisons des 30 échantillons

3.3.1.1 Comparaison entre les classifications moléculaires et botaniques

Dans le tableau 3.2, nous présentons les médianes des indices NMI calculées sur les paires de partitions. Á chaque fois, l'une des partitions et la classification botanique. Dans ce tableau, SW signifie Smith-Waterman et kmers les distances basées sur kmer calculées avec des kmers de longueur $k = 4$.
La première colonne contient les résultats pour les familles sont des valeurs médianes sur dix échantillons, la deuxième colonne contient les résultats pour les genres sont des valeurs médianes sur vingt échantillons et la troisième colonne contient les résultats sur les 30 réplicats.

		Familles		Genres		30 réplicats	
Méthode		SW	kmers	SW	kmers	SW	kmers
CAH	Ward	1	0.61	0.83	0.73	0.87	0.71
	SL	0.88	0.54	0.75	0.59	0.76	0.58
	CL	0.85	0.63	0.75	0.71	0.75	0.67
SBM		0.57	0.52	0.82	0.66	0.68	0.63

Table 3.2 – NMI entre la classification botanique (en familles ou en genres) et les quatre classifications à base moléculaire (ligne) pour deux distances (colonne) et selon que l'on cherche l'accord sur les familles dans les ordres ou sur les genres dans les familles.

Dans le tableau 3.3, nous présentons des statistiquessur les NMIs. Ces statistiques sont calculées sur 30 sous-échantillons (10 ordres et 20 familles).

Paires des méthodes	Min.	1st Qu.	médiane	moyenne	3rd Qu.	Max.
SL vs. botanique	0.013	0.27	0.76	0.64079	1.00	1.00
Ward vs. botanique	0.05	0.40	0.87	0.73	1.00	1.00
CL vs. botanique	0.03	0.47	0.75	0.68	1.00	1.00
SBM vs. botanique	0.10	0.33	0.68	0.64	0.98	1.00

Table 3.3 – Statistiques sur la distribution de l'information mutuelle normalisée (NMI) calculée entre chaque classification moléculaire et la classification botanique. Résultats obtenus en utilisant la distance de Smith-Waterman.

Dans nos résultats, nous observons une tendance commune aux 30 échantillons : quelle que soit le niveau taxonomique ou la distance, les partitions issues de la CAH avec le critère d'agrégation de Ward sont plus proches de la classification botanique. Les autres méthodes se classent différemment en fonction de ces deux variables, mais donnent des partitions assez proches de celles obtenues par la classification botanique. L'accord est meilleur lorsque la classification est faite sur des distances calculées par SW.

3.3.1.2 Comparaison entre les SBM et les trois classifications CAH

Dans le tableau 3.4, nous présentons les statistiques des indices NMI calculés sur les pairs des partitions obtenues par les différentes classifications moléculaires. Dans nos résultats nous montrons une grande proximité entre les six paires de méthodes à part entre SBM et SL. Nous révélons que les SBM donnent des partitions proches de Ward même dans le pire des cas, le NMI minimal est de 0.31.

Paires des méthodes	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
SBM Vs SL	0.039	0.30	0.65	0.59	0.98	1.00
SBM vs. Ward	0.31	0.52	0.74	0.74	1.00	1.00
SBM vs. CL	0.11	0.47	0.65	0.67	0.91	1.00
SL vs. Ward	0.04	0.34	0.78	0.69	1.00	1.00
SL vs. CL	0.05	0.56	0.84	0.74	1.00	1.00
Ward vs CL	0.05	0.70	0.86	0.80	1.00	1.00

TABLE 3.4 – Statistiques sur la distribution de l'information mutuelle normalisée (NMI) calculée entre chaque paire de classifications à base moléculaire. Les statistiques sont calculées sur les NMIs obtenues sur 30 réplicats (10 classifications en familles et 20 en genres).

3.3.2 Jeu de donnée complet

3.3.2.1 Accord entre la classification moléculaire et la classification botanique selon le niveau taxonomique

Dans la figure 3.2, nous présentons l'accord entre les classifications botaniques et moléculaires selon le niveau taxonomique. Chaque courbe correspond à une classification à base moléculaire. Les NMI sont calculées pour les classifications des échantillons associés aux différents niveaux taxonomique.

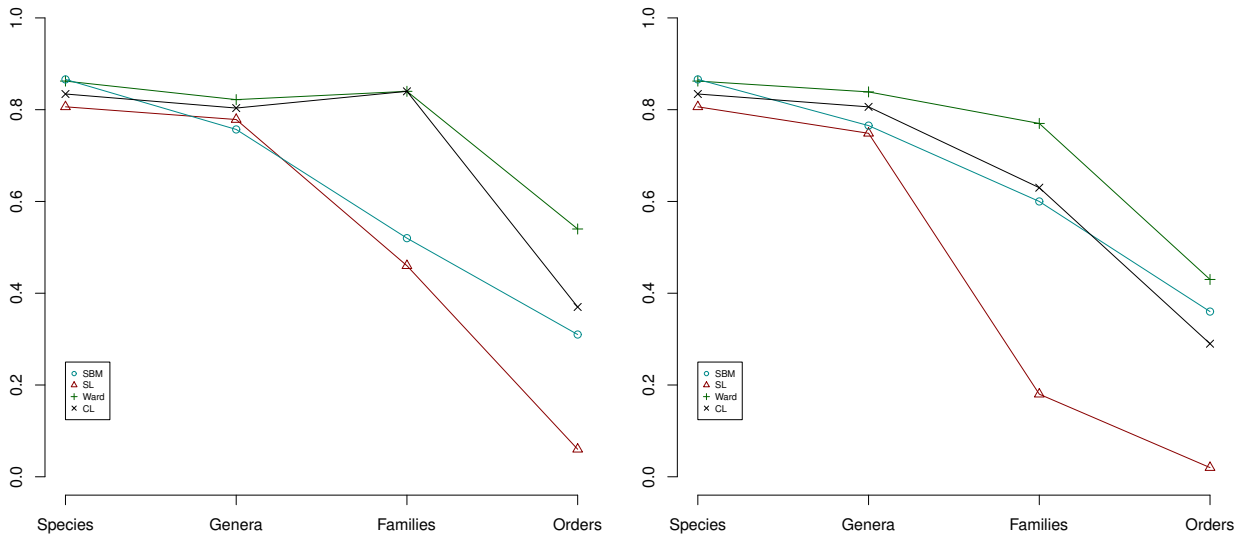


FIGURE 3.2 – Accord entre les classifications botaniques et moléculaires selon le niveau taxonomique, axe x : niveaux taxonomiques, axe y : NMI entre la classification basée sur les molécules et la classification botanique. Dans la figure de gauche : distances Smith-Waterman. Dans la figure de droite : distances basées sur les kmers.

Nous montrons dans nos résultats que dans un premier temps les NMI dépendent du niveau taxonomique niveau taxonomique choisi : plus nous nous approchons des espèces plus les indices sont élevés. Nous révélons ensuite que les trois classifications via SBM et CAH avec Ward et CL sont très proches entre elles et que les quatre méthodes donnent des partitions similaires au niveau des espèces.

3.3.2.2 Diagramme de Sankey

Afin de comprendre les différences de NMI observés sur la figure 3.2, nous regardons en détail les partitions associés à ces classifications. Pour cela, nous utilisons les diagramme de Sankey que nous traçons dans les graphiques 3.3 et 3.4, un flux entre deux classes signifie qu'elles partagent des séquences en commun et la largeur d'un flux entre deux classes est proportionnelle au nombre de séquences appartenant à ces deux classes.

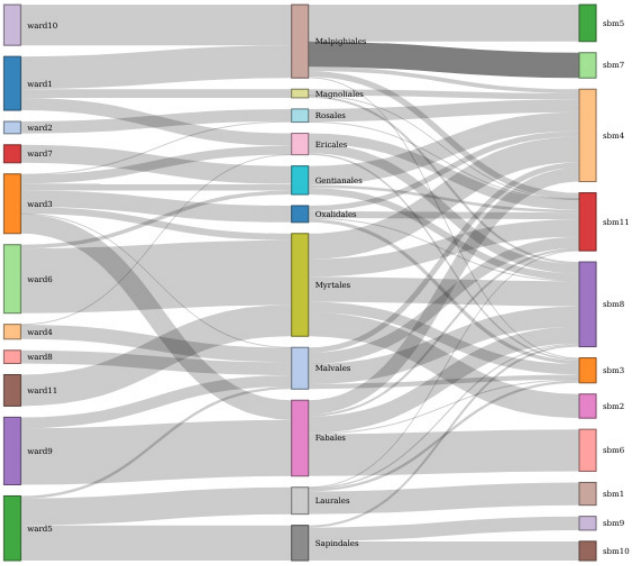


FIGURE 3.3 – Diagramme de Sankey des correspondances entre les partitions issues la CAH avec Ward (colonne de gauche), la classification botanique (colonne centrale) et SBM (colonne de droite) pour les ordres.

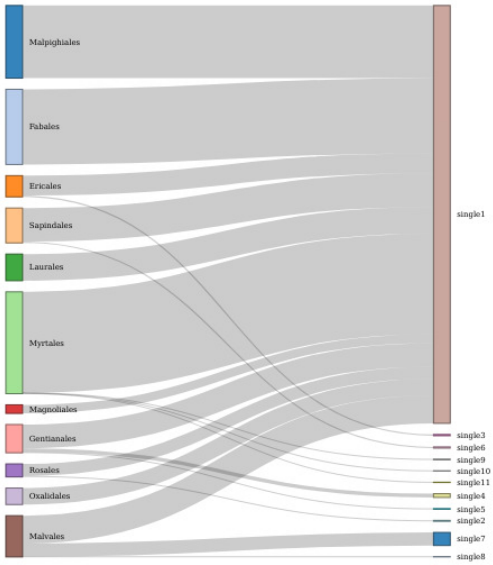


FIGURE 3.4 – Diagramme de Sankey des correspondances entre les partitions issues la classification botanique (colonne de gauche) et la CAH avec SL (colonne de droite) pour les ordres.

Sur la figure 3.3, un ordre est divisé en plusieurs classes dans les classifications Ward ou SBM ou, au contraire, une classe Ward ou SBM est composé d'individus issus de plusieurs classes botaniques. Ce dernier cas est plus problématique lors de l'interprétation des classifications basées sur les distances moléculaires.
Sur la figure 3.4, nous observons que l'une des classes SL est formée par presque tous les ordres présents dans l'ensemble de données. Cela explique le faible NMI pour SL pour une classification au niveau des ordres.

3.4 Conclusion

Dans ce chapitre, nous avons comparé les méthodes de classifications moléculaires entre elles et avec la classification botanique pour des niveaux taxonomiques élevés et nous avons montré qu'il y a un bon accord entre les deux. De plus, nous avons nous avons montré que la classification SBM permet de détecter, par le biais de la distribution estimée des distances intra-classe, des situations où les classifications moléculaires peuvent être en désaccord avec la classification botanique parce que la matrice de distance n'est pas clairement structurée en communautés.
Un autre avantage des SBM comparés à la CAH est qu'en plus de nous fournir les partitions, les SBM produisent la matrice de distances inter et intra classes Λ. Cette matrice fournis une information très importante comme présenté dans l'introduction.

Chapitre 4

Approche "Tenseur-Train" pour la séparation des variables dans les modèles graphiques

Table des matières

4.1 Introduction

Dans ce chapitre, nous rappelons quelques notions d'algèbre [Str19, HJ12, Ste93] puis nous décrivons l'approche de présentée par Novikov [NROV14] pour l'approximation d'une loi jointe d'un modèle graphique par un tenseur au format "Tenseur-Train". L'avantage de cette approche est de conduire naturellement à la séparation des variables. Le tenseur est simplement le tableau multivarié des probabilités indexées par les différents états de la loi jointe du modèle graphique. Nous montrerons dans le chapitre 5 que si les liens sont au plus binaires dans le modèle graphique, comme dans un modèle SBM, alors le tenseur peut être écrit exactement dans un format "Tenseur-Train". Nous utiliserons cette approche pour le calcul des marginales d'un modèle à blocs stochastiques.

4.2 Rappel de quelques bases d'algèbre linéaire

Durant cette section, nous rappelons quelques définitions d'algèbre linéaire que nous utiliserons dans les sections 4.4.4 et 7.3.

Norme de Frobenius. Soit $A \in \mathbb{R}^{p\times q}$ une matrice de coefficients $(a_{i,j})_{i,j\in[\![1,p]\!]\times[\![1,q]\!]}$. La norme de Frobenius est définie comme :

$$\|A\|_F = \sqrt{\sum_{i=1}^{p}\sum_{j=1}^{q} a_{i,j}^2} \tag{4.2.1}$$

Pour alléger les notations, nous notons la norme de Frobenius $\|.\|$.

Produit de Kronecker. Soient deux matrices $A \in \mathbb{R}^{n,m}$ et $B \in \mathbb{R}^{p,q}$. Le produit de Kronecker noté $\otimes_k$ entre A et B est une matrice de taille $np \times mq$ dont les coordonnées sont :

$$A \otimes_k B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & \dots & a_{1m}B \\ a_{21}B & \ddots & \dots & \dots & \vdots \\ \vdots & \dots & \ddots & \dots & \vdots \\ \vdots & \dots & a_{ij}B & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1}B & \vdots & \vdots & \vdots & a_{nm}B \end{pmatrix} \tag{4.2.2}$$

Le produit de Kronecker $\otimes_k$ vérifie les deux propriétés suivantes :

- $\mathbf{P_1}$: "Mixed-product property" : Si A, B, C et D sont quatre matrices de tailles telles que les produits AC et BD ont un sens, alors :

$$AC \otimes_k BD = (A \otimes_k B)(C \otimes_k D) \tag{4.2.3}$$

- $\mathbf{P_2}$: Si $x, y \in \mathbb{R}$ alors $x \otimes_k y \in \mathbb{R}$ et :

$$x \otimes_k y = xy \tag{4.2.4}$$

qui seront utilisées dans ce chapitre.

Matrices orthogonales : Soit $n \in \mathbb{N}$ et $A \in \mathbb{R}^{n \times n}$
A est une matrice orthogonale si :

$$A^T A = A A^T = \mathbb{I}_n \tag{4.2.5}$$

On distingue une matrice orthogonale qui est carrée d'une matrice rectangulaire dont les colonnes sont orthogonales.

Factorisation QR [Gen98] Soit une matrice $A \in \mathbb{R}^{p \times q}$. La décomposition QR de cette matrice est l'écriture de A sous-forme d'un produit d'une matrices Q dont les colonnes sont orthogonales et d'une matrice R triangulaire supérieure selon :

$$A = QR \tag{4.2.6}$$

Remarque : Dans le cas où A est une "Fat Matrix" [Bay13], c'est-à-dire son nombre de lignes p est inférieur à son nombre de colonnes q, nous définissons une décomposition LQ où : Q est une matrice dont les colonnes sont orthogonales et L est une matrice triangulaire inférieure. Cette décomposition sera utilisée durant la phase d'orthogonalisation des cores dans l'algorithme du rounding présenté dans la section 4.4.4.

Décomposition en valeur singulière (singular value decomposition en anglais, d'où l'abbréviation SVD) [WRR03, Str19, HJ12] .
Soient $p, q \in \mathbb{N}$ avec $p \geq q$, $A \in \mathbb{R}^{p \times q}$, alors, il existe $U \in \mathbb{R}^{p \times q}$ et $V \in \mathbb{R}^{q \times q}$ dont les colonnes sont orthogonales et $\Sigma \in \mathbb{R}^{q \times q}$ une matrice diagonale telle que :

$$A = U \Sigma V^T \tag{4.2.7}$$

la matrice Σ s'exprime comme $\Sigma = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_q)$. σ_i ses éléments vérifient : $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_p \geq 0$

Cette décomposition est exacte comme expliquée dans l'article [MIR60].
Soit,

$$A = U \Sigma V^T$$

Nous rappelons que

$$\|A\|^2 = Tr(A^T A)$$

d'où

$$\|A\|^2 = Tr(V \Sigma U^T U \Sigma V^T)$$

Comme les matrices U et V sont orthogonales, $U^T U = I_p$, et nous avons

$$Tr(V \Sigma U^T U \Sigma V^T) = Tr(V \Sigma^2 V^T) = Tr(\Sigma^2)$$

car la trace est invariante par rotation. Nous en déduisons,

$$\|A\|^2 = Tr(\Sigma^2) = \sum_{k=1}^{q} \sigma_k^2$$

Approximation de rang faible : Soit $r < q$. Nous pouvons séparer la somme $\sum_{k=1}^{q} \sigma_k^2$ en deux parties :

$$\|A\|^2 = \sum_{k=1}^{r} \sigma_k^2 + \sum_{k=r+1}^{q} \sigma_k^2 \tag{4.2.8}$$

Si nous retenons les r premières valeurs singulières, nous pouvons définir Σ_r comme, $\Sigma_r = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_r)$. Puis nous retenons les r premières colonnes $(u_1, ..., u_r)$ (resp. $(v_1, ..., v_r)$) de U (resp. de V) pour définir les matrices $U_r \in \mathbb{R}^{p \times r}$ (resp. $V_r \in \mathbb{R}^{q \times r}$).
Ensuite, nous définissons :

$$A_r = U_r \Sigma_r V_r^T \tag{4.2.9}$$

La norme de A_r s'exprime :

$$\|A_r\|^2 = \sum_{k=1}^{r} \sigma_k^2 \tag{4.2.10}$$

Le théorème d'Eckart-Young [EY36] stipule que la matrice A_r est la matrice de rang r qui minimise $\|A - A_r\|$ selon la norme de Frobenius. C'est-à-dire :

$$\|A - A_r\| = \min_{B \in \mathbb{R}^{p \times q}, \text{rang}(B) = r} (\|A - B\|) \tag{4.2.11}$$

Il est possible de faire une approximation de la matrice A par la meilleure approximation de rang faible ("best low rank approximation" (BLRA) [HJ12, Ste93]) de deux manières :

Approximation de rang faible à rang prescrit : Cette approche consiste à fixer le rang r et calculer A_r. La précision de A_r est donnée par le théorème d'Eckart-Young.
Nous injectons l'expression de la norme de A_r (4.2.10) dans la formule (4.2.8) et nous obtenons :

$$\|A\|^2 = \sum_{k=1}^{r} \sigma_k^2 + \sum_{k=r+1}^{q} \sigma_k^2 = \|A_r\|^2 + \sum_{k=r+1}^{q} \sigma_k^2 \tag{4.2.12}$$

Ainsi, l'erreur de l'approximation à rang prescrit r s'exprime

$$\|E\|^2 = \|A\|^2 - \|A_r\|^2 = \sum_{k=r+1}^{q} \sigma_k^2 \tag{4.2.13}$$

Approximation à précision prescrite ϵ : Cette approche consiste à trouver le rang r minimal tel que si A_r est la meilleure approximation de A de rang r, alors l'erreur vérifie : $\|E\| \leq \epsilon \|A\|$. Soit,

$$\sum_{k=r+1}^{q} \sigma_k^2 \leq \epsilon \left(\sum_{k=1}^{q} \sigma_k^2 \right) \tag{4.2.14}$$

Ensuite, nous calculons :

$$A_r = U_r \Sigma_r V_r^T \tag{4.2.15}$$

Lorsque nous faisons un SVD à précision prescrite, nous ne pouvons pas contrôler le rang de la matrice A_r. Quand nous faisons une SVD avec un rang prescrit, nous ne pouvons pas contrôler l'erreur d'approximation.

4.3 Format "Tenseur Train"

Un objet d'étude dans cette thèse est la loi jointe d'un modèle SBM avec n individus écrite sous forme d'un tenseur d'ordre n, pour utiliser des résultats de calcul tensoriel et proposer de nouveaux algorithmes pour le calcul des marginales. Pour cela, nous montrerons que la loi jointe d'un modèle SBM conditionnellement aux distances peut s'écrire exactement comme un tenseur en format Tenseur-Train (TT). Aussi, nous rappelons dans cette section quelques résultats connus sur ce format. Nous adoptons les notations présentées par Oseledets dans les articles [OD12] et [OTZ11]

4.3.1 La loi jointe d'un modèle SBM comme un tenseur

Les tenseurs sont définis algébriquement comme des formes multilinéaires [Fra22b]. Ils peuvent être aussi considérés de façon moins abstraite comme des tableaux multidimensionnels, c'est-à-dire des généralisations des scalaires, des vecteurs et des matrices. Un tel tableau est l'écriture d'un tenseur dans une base choisie pour chacun des espaces vectoriels sur lesquels la forme multilinéaire opère.
Un tenseur d'ordre $n \in \mathbb{N}$ est un tableau n dimensionnel tel que chaque coefficient s'exprime avec n indices. Nous pouvons ainsi écrire la loi jointe $\mathbf{\Psi}[i_1, \ldots, i_n] \in \mathbb{R}$, d'un SBM comme un tenseur $\mathbf{\Psi}$, où $\mathbf{\Psi}[i_1, \ldots, i_n] \in \mathbb{R}$, est l'élément d'indice $(i_1, \ldots, i_n)$ de $\mathbf{\Psi}$ (indice $n-$dimensionnel). Dans ce cas :

- n est appelé l'ordre du tenseur $\mathbf{\Psi}$.
- les rangs des indices $(1, \ldots, n)$ sont les modes du tenseur
- i_k est l'indice pour le mode k
- si $i_k \in [\![1, q_k]\!]$, alors q_k est la dimension du tenseur pour l'ordre k. La dimension multilinéaire du tenseur est $(q_1, \ldots, q_n)$.

Dans notre cas, pour un tenseur associé à un SBM :

→ toutes les dimensions q_k sont égales à Q.

→ le nombre d'individus, n, est l'ordre du tenseur,

→ chaque individu définit un mode.

→ Q, le nombre de classes, est la dimension de chaque mode

Notons que, souvent, dans la littérature, le mot dimension peut désigner l'ordre. Nous utiliserons le mot ordre dans ce document.

4.3.2 Contraction des tenseurs

Dans cette section, nous rappelons sur quelques exemples la définition d'une contraction entre deux tenseurs. Un exemple de contraction est le produit matriciel, où chaque matrice est lue comme un tenseur d'ordre 2 : si $C = AB$, alors

$$\left.\begin{array}{l} A = [a_{ij}]_{i,j} \\ B = [b_{jk}]_{j,k} \end{array}\right\} \implies C = [c_{ik}]_{i,k} \quad \text{avec} \quad c_{ik} = \sum_j a_{ij} b_{jk} \tag{4.3.1}$$

Chaque matrice a deux modes, repérés par leurs indces : i, j pour A, j, k pour B. Il y a un mode commun : j. La contraction est simplement la somme des produits des coefficints de A et B selon le mode commun k. Cela se généralise à plus de deux modes, avec sur un exemple qui reste simple : soit $\mathbf{A} = [a_{ijk}]_{i,j,k}$ un tenseur d'ordre 3 d'éléments a_{ijk} et $\mathbf{B} = [b_{k\ell}]_{k,\ell}$ un tenseur

d'ordre 2 d'éléments $b_{k\ell}$. Les tenseurs $\mathbf{A}$ et $\mathbf{B}$ partagent un indice commun k et l'opération de contraction est la somme des produits des coefficients sur cet indice. Elle est notée $\bullet_k$. Aussi

$$\mathbf{A} = [a_{ijk}]_{i,j,k}, \qquad \mathbf{B} = [b_{k\ell}]_{k,\ell} \quad \Longrightarrow \quad \mathbf{C} = \mathbf{A} \bullet_k \mathbf{B} = \left[\sum_k a_{ijk} b_{k\ell} \right]_{i,j,\ell} \tag{4.3.2}$$

Le tenseur contracté $\mathbf{C}$ résultant est d'ordre 3. Cette formule se généralise aisément pour la contraction de deux tenseurs d'ordre quelconques sur leurs modes communs (il peut y en avoir plus d'un, comme par exemple dans

$$c_{ijmpq} = \sum_{k,\ell} a_{ijk\ell m} b_{pk\ell q}$$

).

4.3.3 Formalisme "Tenseur-Train"

Ce format a été proposé par l'équipe d'Oseledets & al. dans les articles [Ose09, OTZ11, OD12].

4.3.3.1 Expression d'un tenseur sous format "Tenseur-Train"

Une manière efficace de représenter $\mathbf{\Psi}$ est de l'écrire avec le formalisme tenseur train lorsque c'est possible. Lorsque ce n'est pas possible, nous recherchons la meilleure approximation de $\mathbf{\Psi}$ par un tenseur $\widetilde{\mathbf{\Psi}}$ en format Tenseur-Train avec un rang donné. Dans le format TT, chaque coefficient de $\mathbf{\Psi}$ est décomposé comme un produit de matrices. Nous le montrons d'abord sur un petit exemple pour un tenseur d'ordre 4. Ses quatre modes sont représentés par les indices i, j, k, ℓ. Chaque coefficient sera un produit de quatre matrices, une par mode, selon

$$\mathbf{\Psi}[i,j,k,\ell] = G_1(i) \cdot G_2(j) \cdot G_3(k) \cdot G_4(\ell) \quad \text{avec} \quad \begin{cases} G_1(i) \in & \mathbb{R}^{1 \times r_1} \\ G_2(j) \in & \mathbb{R}^{r_1 \times r_2} \\ G_3(k) \in & \mathbb{R}^{r_2 \times r_3} \\ G_4(\ell) \in & \mathbb{R}^{r_3 \times 1} \end{cases} \tag{4.3.3}$$

Les entiers r_1, r_2, r_3 sont appelés les TT-rangs du format TT. Le plus grand des trois est appelé le TT-rang maximal de $\mathbf{\Psi}$ au format TT [NROV14].

On peut noter que, par exemple, $G_1(i)$ pour $\mathbf{\Psi}[i,j,k,\ell]$ ne dépend ni de j, ni de k, ni de ℓ. Par exemple, nous avons également

$$\mathbf{\Psi}[i,j',k',\ell'] = G_1(i) \cdot G_2(j') \cdot G_3(k') \cdot G_4(\ell') \tag{4.3.4}$$

Les premier et dernier éléments sont des vecteurs et les autres sont des matrices. Dans cette écriture, chaque coefficient du tenseur $\mathbf{\Psi}$ est obtenu par des produits matriciels où un vecteur horizontal ($G_1(i)$) est une matrice à une ligne et un vecteur vertical ($G_4(\ell)$) est une matrice à une colonne. L'expression de chaque coefficient comme un produit de matrices peut s'illustrer selon

$$\mathbf{\Psi}[i,j,k,\ell] = \underline{\qquad} \cdot \boxed{} \cdot \boxed{} \cdot \Big|$$

Il est possible de regrouper par exemple l'ensemble des matrices $(G_2(j))_j$ comme tranches d'un tenseur d'ordre 3 noté $\mathbf{G}_2$, avec trois modes de dimensions respectives r_1, Q, r_2. Donc

$\mathbf{G}_2 \in \mathbb{R}^{r_1 \times Q \times r_2}$. Cela peut se faire pour chaque mode, pour donner :

$$\begin{cases} \mathbf{G}_1 \in & \mathbb{R}^{1 \times Q \times r_1} \\ \mathbf{G}_2 \in & \mathbb{R}^{r_1 \times Q \times r_2} \\ \mathbf{G}_3 \in & \mathbb{R}^{r_2 \times Q \times r_3} \\ \mathbf{G}_4 \in & \mathbb{R}^{r_3 \times Q \times 1} \end{cases} \tag{4.3.5}$$

les tenseurs $\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3, \mathbf{G}_4$ sont applés les "cores" du format TT du tenseur $\mathbf{\Psi}$.
À partir du format tensoriel des cores présenté dans la formule (4.4.7), nous pouvons écrire la formule (4.3.3) en remplaçant les produits matriciels par des contractions de tenseurs $\mathbf{G}_k$:

$$\mathbf{\Psi} = \mathbf{G}_1 \bullet_\alpha \mathbf{G}_2 \bullet_\beta \mathbf{G}_3 \bullet_\gamma \mathbf{G}_4 \tag{4.3.6}$$

soit, en développant

$$\mathbf{\Psi}[i,j,k,\ell] = \sum_{\alpha=1}^{r_1} \sum_{\beta=1}^{r_2} \sum_{\gamma=1}^{r_3} \mathbf{G}_1(i,\alpha)\mathbf{G}_2(\alpha,j,\beta)\mathbf{G}_3(\beta,k,\gamma)\mathbf{G}_4(\gamma,\ell) \tag{4.3.7}$$

Les formules (4.3.3) et (4.3.7) se généralisent pour des tenseurs $\mathbf{\Psi}$ d'ordres supérieurs à 4.

4.3.3.2 Avantage de ce format

Le format TT présente plusieurs avantages. Nous les détaillons dans cette section.

Séparation des variables : Dans $\mathbf{\Psi}[i,j,k,\ell]$, $G_1(i)$ ne dépend que de i, $G_2(j)$ ne dépend que de j ... Cela conduit naturellement à la séparation des variables. En effet : la constante de normalisation s'écrit, sur cet exemple :

$$\begin{aligned} W &= \sum_{i,j,k,l} \mathbf{\Psi}[i,j,k,\ell] \\ &= \sum_{i,j,k,l} G_1(i).G_2(j).G_3(k).G_4(\ell) \\ &= \left(\sum_i G_1(i)\right)\left(\sum_j G_2(j)\right)\left(\sum_k G_3(k)\right)\left(\sum_\ell G_4(\ell)\right) \end{aligned} \tag{4.3.8}$$

car la multiplication est distributive vis à vis de l'addition dans l'anneau des matrices. C'est l'écriture de W en séparant les variables i,j,k,ℓ. Nous utiliserons cette séparation des variables plus généralement dans la suite pour le calcul de différentes marginales.

Réduction du stockage : Stocker le tenseur $\mathbf{\Psi}$ nécessite un stockage de Q^n éléments. Si n est grand, le stockage de ce tenseur devient impossible. L'utilisation du format tenseur train avec le produit matriciel, par exemple ici pour $n = 4$, permet de stocker les éléments sous forme de $2Q$ matrices $r \times r$ (en supposant que $r_1 = r_2 = r_3 = r$), Q matrices $G_2(j)$ pour $1 \leq j \leq Q$ et Q matrices $G_3(k)$ pour $1 \leq k \leq Q$ d'une part, et de $2Q$ vecteurs de $\mathbb{R}^r$ d'autres parts. Au total, cela donne $2Qr + 2Qr^2 = 2Qr(r+1)$ élements soit en $\mathcal{O}(2Qr^2)$, au lieu de Q^4. Le coût du stockage en format TT pour un tenseur d'ordre quelconque n se généralise facilement en $\approx \mathcal{O}(nQr^2)$. Grâce à ce format, nous passons d'un stockage exponentiel en n des tenseurs à un stockage linéaire en n.

Opérations élémentaires dans le format TT : Les opérations tensorielles élémentaires (sommes, produits, multiplication par un scalaire ...) peuvent être exprimées de manière exacte dans le format TT sans avoir à expliciter les coefficients du tenseur comme présenté dans [Ose11]. Nous présentons dans la section 4.4 une adaptation de ce format vers des objets appelés TT-matrices et TT-vecteurs, et les opérations plus complexes de produit matrice $\times$ matrice ou matrice $\times$ vecteur dans ce format.
Par exemple, si $n = 3$, nous donnons deux tenseurs en format TT selon

$$\begin{cases} \mathbf{A}[i,j,k] &= G_1^{\mathbf{A}}(i).G_2^{\mathbf{A}}(j).G_3^{\mathbf{A}}(k) \\ \mathbf{B}[i,j,k] &= G_1^{\mathbf{B}}(i).G_2^{\mathbf{B}}(j).G_3^{\mathbf{B}}(k) \end{cases} \tag{4.3.9}$$

Alors, leur somme $\mathbf{D} = \mathbf{A} + \mathbf{B}$ s'exprime en format TT selon

$$\mathbf{D}[i,j,k] = G_1^{\mathbf{D}}(i).G_2^{\mathbf{D}}(j).G_3^{\mathbf{D}}(k) \tag{4.3.10}$$

avec

$$\begin{cases} G_1^{\mathbf{D}}(i) &= \begin{pmatrix} G_1^{\mathbf{A}}(i) & G_1^{\mathbf{B}}(i) \end{pmatrix} \\ G_2^{\mathbf{D}}(j) &= \begin{pmatrix} G_2^{\mathbf{A}}(j) & 0 \\ 0 & G_2^{\mathbf{B}}(j) \end{pmatrix} \\ G_3^{\mathbf{D}}(k) &= \begin{pmatrix} G_3^{\mathbf{A}}(k) \\ G_3^{\mathbf{B}}(k) \end{pmatrix} \end{cases} \tag{4.3.11}$$

comme nous pouvons le vérifier en effectuant les produits matriciels :

$$D[i,j,k] = G_1^{\mathbf{A}}(i).G_2^{\mathbf{A}}(j).G_3^{\mathbf{A}}(k) + G_1^{\mathbf{B}}(i).G_2^{\mathbf{B}}(j).G_3^{\mathbf{B}}(k) \tag{4.3.12}$$

Cette formule peut se généraliser pour des tenseurs d'ordre $n \in \mathbb{N}$: Si $\mathbf{A}$ et $\mathbf{B}$ sont des tenseurs d'ordre n dont les cores respectifs sont $\mathbf{G}_k^{\mathbf{A}}(i_k)$ et $\mathbf{G}_k^{\mathbf{B}}(i_k)$, alors $\mathbf{D} = \mathbf{A} + \mathbf{B}$ s'exprime en format TT selon :

$$\mathbf{D}[i_1, i_2, \ldots, i_d] = \mathbf{G}_1^{\mathbf{D}}(i_1).\mathbf{G}_2^{\mathbf{D}}(i_2).\ldots.\mathbf{G}_n^{\mathbf{D}}(i_n) \tag{4.3.13}$$

avec

$$\begin{cases} k = 1 & \mathbf{G}_1^{\mathbf{D}}(i_1) &= \begin{pmatrix} \mathbf{G}_1^{\mathbf{A}}(i_1) & \mathbf{G}_1^{\mathbf{B}}(i_1) \end{pmatrix} \\ \forall k \in [\![2, n-1]\!] & \mathbf{G}_k^{\mathbf{D}}(i_k) &= \begin{pmatrix} \mathbf{G}_k^{\mathbf{A}}(i_k) & 0 \\ 0 & \mathbf{G}_k^{\mathbf{B}}(i_k) \end{pmatrix} \\ k = n & \mathbf{G}_n^{\mathbf{D}}(i_n) &= \begin{pmatrix} \mathbf{G}_n^{\mathbf{A}}(i_n) \\ \mathbf{G}_n^{\mathbf{B}}(i_n) \end{pmatrix} \end{cases} \tag{4.3.14}$$

Nous expliquerons dans la section 4.4.3.2 comment faire les produits matriciels dans ce format.

4.3.4 Tensor Network

Les "Tensor network" sont une famille de contractions sur un ensemble de tenseurs qui partagent certains modes. Il s'agit de factorisations de très grands tenseurs en réseau de plus petits tenseurs.
Prenons l'exemple d'un produit matriciel. Soit $A \in \mathbb{R}^{n \times p}$ et $B \in \mathbb{R}^{p \times m}$. Le produit matriciel $C = A.B$ s'exprime $c_{ik} = \sum_{j=1}^{p} a_{ij} b_{jk}$. Ce produit peut être représenté en tenseur Network selon

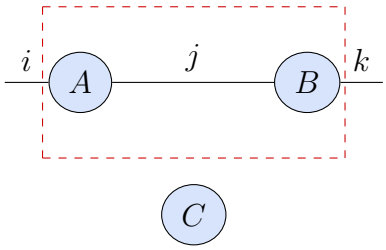


FIGURE 4.1 – Représentation Tensor network d'un produit matriciel

le schéma 4.1 : Les nœuds représentent les matrices A et B, le lien entre eux est l'indice commun j sur lequel nous faisons la somme. Les indices i (resp. k) propre à la matrice A (resp. B) sont les liens vers l'extérieur.
Nous pouvons exprimer ce produit matriciel comme une contraction selon l'indice j :

$$C = A \bullet_j B \tag{4.3.15}$$

Nous nous intéressons à la représentation Tensor Network d'un "Tenseur Train" [LC15] où nous représentons le tenseur $\mathbf{\Psi}$ exprimé dans la formule (4.3.3) dans le schéma 4.2 :

FIGURE 4.2 – Représentation Tensor network de type Tenseur Train

Dans ce schéma, les nœuds représentent les cores $(\mathbf{G}_k)_{k\in[\![1,n]\!]}$ dans l'expression au format TT de $\mathbf{\Psi}$. Les liens entre deux nœuds $(\alpha_k)_{k\in[\![1,n-1]\!]}$ représentent les indices partagés entre deux cores. Enfin, les liens vers le haut représentent les indices libres des cores.

Cette représentation visualise l'écriture algébrique

$$\mathbf{\Psi} = \mathbf{G}_1 \bullet_{\alpha_1} \mathbf{G}_2 \bullet_{\alpha_2} \dots \bullet_{\alpha_{n-1}} \mathbf{G}_n \tag{4.3.16}$$

Cette visualisation sera utilisée dans le cadre des TT-vecteurs (dans le paragraphe 4.4.2) et des TT-matrices (dans le paragraphe 4.4.2).

En plus des Tenseur Trains, il existe dans la littérature plusieurs autres types de Tensor Network, représentés sous forme de graphes. Parmi les tenseurs networks connus, nous pouvons citer : Matrix Product State / Tenseur Train, Tree Tensor Network / Hierarchical Tucker, PEPS, MERA ... Dans le schéma4.3, nous montrons la représentation de ces différents types de Tensor Network :

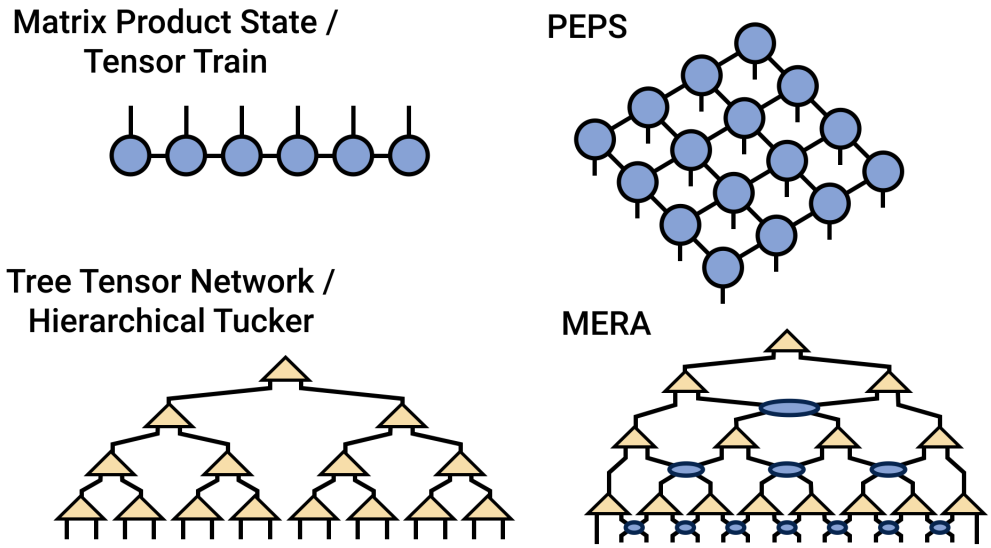


FIGURE 4.3 – Tensor networks (figure issue de https://tensornetwork.org/)

Les articles [Cic14], [Orú14] et [Orú19], détaillent les différents formalismes Tensor Network. Á noter que les Tenseur Train sont des cas particuliers de Three tensor Network.

4.4 Format TT-matrices

Dans cette section, nous précisons la notion de multiplexage, donnons ensuite les définitions des objets TT-matrices/TT-vecteurs, présentons les opérations dans ce format, et enfin nous expliquerons comment faire la compression des TT-matrices grâce à l'algorithme du TT-rounding.

4.4.1 Notions de multiplexage et de démultiplexage

Soient $n_1, \ldots, n_d \in \mathbb{N}$ et $n \in \mathbb{N}$ avec $n = \prod_{k=1}^{d} n_k$.
Nous notons :

$$I_N = [\![1, n]\!], \qquad \forall\, k,\, I_k = [\![1, n_k]\!] \tag{4.4.1}$$

Le multiplexage est défini comme une fonction bijective ρ qui fait correspondre un indice unique i à un multi indice $\mathbf{i}$:

$$\mathbf{i} = (i_1, \ldots, i_d) \in I_1 \times \ldots \times I_d \xrightarrow{\rho} i \in I_N \tag{4.4.2}$$

On note

$$\begin{cases} \mathbf{i} = (i_1, \ldots, i_d) & \in I_1 \times \ldots \times I_d \quad \text{Forme démultiplexée} \\ i = \rho(\mathbf{i}) = i_d + \sum_{k=1}^{d-1} n_k (i_k - 1) & \in I_N \quad \text{Forme multiplexée} \end{cases} \tag{4.4.3}$$

La formule (4.4.3) peut se démontrer par une récurrence.

Nous utiliserons ces notions pour faire la correspondance entre les matrices et les tenseurs dans les différentes définitions qui vont suivre. L'indice unique i correspondra aux indices des lignes et colonnes d'une TT-matrice, et chaque indice i_k du multi-indice $\mathbf{i}$ à un mode d'un tenseur qui lui est équivalent.

4.4.2 Définitions de base

Dans cette section, nous expliquons comment faire le passage des matrices et vecteurs au format TT grâce au multiplexage. Nous considérons $n, m \in \mathbb{N}$ et $d \in \mathbb{N}$.

TT-vecteurs Soient $r_0, \dots r_d \in \mathbb{N}^d$, $n_1, \dots n_d \in \mathbb{N}^d$, et $n = \prod_{k=1}^d n_k$. Un vecteur $X \in \mathbb{R}^n$ est exprimé dans le format TT-vecteur [NPOV15a] sachant

$$i = \rho(i_1, \dots, i_d) \tag{4.4.4}$$

si

$$\forall i \in [\![1, n]\!], \quad X[i] = \mathbf{X}[i_1, \dots, i_d] = G_1(i_1).G_2(i_2).\dots.G_d(i_d)$$
$$\text{avec} \quad \begin{cases} k = 1 & \Rightarrow \quad G_1(i_1) \quad \in \mathbb{R}^{r_0 \times r_1} \\ 1 < k < d & \Rightarrow \quad G_k(i_k) \quad \in \mathbb{R}^{r_{k-1} \times r_k} \\ k = d & \Rightarrow \quad G_d(i_d) \quad \in \mathbb{R}^{r_{d-1} \times r_d} \end{cases} \tag{4.4.5}$$

avec $r_0 = r_d = 1$. Dans cette formule, la correspondance entre un vecteur X et un tenseur $\mathbf{X}$ est faite par démultiplexage : X (TT-vecteur) est la forme multiplexée et $\mathbf{X}$ (tenseur) est la forme démultiplexée. Par extension, il est convenu d'appeler également le tenseur $\mathbf{X}$ TT-vecteur. Nous appelons $(r_0, \dots, r_d)$ le rang multilinéaire ou TT-rang du TT-vecteur $\mathbf{X}$ et $r^{\mathbf{X}} = \max(r_1, \dots, r_{d-1})$ son TT-rang maximal.

Ce TT-vecteur est un tensor network selon la figure 4.4 :

FIGURE 4.4 – Tensor network d'un TT-vecteur $\mathbf{X}$

Les figures 4.4 et 4.2 sont similaires. Elles représentent la représentation d'un TT-vecteur sous format démultiplexé. En développant les contractions des cores, nous obtenons :

$$\mathbf{x}[i_1, \dots, i_d] = \sum_{\alpha_1=1}^{r_1} \dots \sum_{\alpha_{d-1}=1}^{r_{d-1}} \mathbf{G}_1(i_1, \alpha_1)\mathbf{G}_2(\alpha_1, i_2, \alpha_2) \dots \mathbf{G}_d(\alpha_{d-1}, i_d) \tag{4.4.6}$$

Comme pour les tenseurs en format TT, nous pouvons regrouper les $(G_k(i_k))_{i_k \in [\![1, n_k]\!]}$ dans un tenseur

$$\begin{cases} k = 1 & \Rightarrow \quad \mathbf{G}_1 \in \quad \mathbb{R}^{1 \times n_1 \times r_1} \\ 1 < k < d & \Rightarrow \quad \mathbf{G}_k \in \quad \mathbb{R}^{r_{k-1} \times n_k \times r_k} \\ k = d & \Rightarrow \quad \mathbf{G}_d \in \quad \mathbb{R}^{r_{d-1} \times n_d \times 1} \end{cases} \tag{4.4.7}$$

Ce format permet de stocker un vecteur de taille $\prod_{k=0}^d n_k$ en $\sum_{k=1}^d n_k r_k r_{k-1}$. Si $n_0 = \dots = n_d = Q$ et $r_1 = \dots = r_{d-1} = r$, le stockage sera en $\mathcal{O}((d-2)Qr^2)$ au lieu de Q^d.

TT-matrices Soient $r_0, \dots r_d \in \mathbb{N}^d$, $n_0, \dots n_d \in \mathbb{N}^d$ et $m_0, \dots m_d \in \mathbb{N}^d$ avec $n = \prod_{k=1}^d n_k$ et $m = \prod_{k=1}^d m_k$. Une matrice $A \in \mathbb{R}^{n \times m}$ est exprimée dans un format TT-matrice [OTZ11], sachant

$$i = \rho(i_1, \dots, i_d), \qquad j = \rho(j_1, \dots, j_d) \tag{4.4.8}$$

si

$$\forall i \in [\![1,n]\!], j \in [\![1,m]\!], \quad A[i,j] = \mathbf{A}[i_1,\dots,i_d\,;j_1,\dots,j_d] = \mathbf{G}_1(i_1,j_1).\mathbf{G}_2(i_2,j_2).\dots.\mathbf{G}_d(i_d,j_d)$$
$$\text{avec} \quad \begin{cases} k=1 & \Rightarrow \quad \mathbf{G}_1(i_1,j_1) \quad \in \mathbb{R}^{r_0\times r_1} \\ 1<k<d & \Rightarrow \quad \mathbf{G}_k(i_k,j_k) \quad \in \mathbb{R}^{r_{k-1}\times r_k} \\ k=d & \Rightarrow \quad \mathbf{G}_d(i_d,j_d) \quad \in \mathbb{R}^{r_{d-1}\times r_d} \end{cases} \tag{4.4.9}$$

Avec $r_d = r_0 = 1$. Dans cette formule, la correspondance entre une matrice A et un tenseur $\mathbf{A}$ avec double indexage est faite par démultiplexage : A est la forme multiplexée et $\mathbf{A}$ est la forme démultiplexée. Par extension, nous appelons le tenseur $\mathbf{A}$ TT-matrice. Nous pouvons passer de $\mathbf{A}$ (resp. $\mathbf{X}$) à A (resp. X) par un reformatage (reshape).
Oseledets a prouvé l'existence de cette décomposition pour toute matrice de taille $2^d \times 2^d$ dans l'article [Ose10]. Il est possible de généraliser cette écriture pour des matrices quelconques, cependant cela conduira à des cores de rangs fort. En effet, si nous considérons que les cores $\mathbf{G}_k(i_k,j_k)$ sont tous de rang r, ce sont des matrices de tailles $r \times r$. Nous aurons n matrices de tailles $r \times r$ et comme chaque indice i_k où j_k varie entre 1 et Q alors nous aurons nr^2Q^2 termes. Donc,

$$Q^{2n} = nr^2Q^2$$

Alors : $r = \frac{Q^{n-1}}{\sqrt{n}}$.

TT-rang maximal d'une TT-matrice Nous appelons TT-rang maximal d'une TT-matrice $\mathbf{A}$ le maximum des TT-rangs de ses cores, c'est-à-dire :

$$\max \text{TT-rang}(\mathbf{A}) = \max(r_1,\dots,r_{d-1}) \tag{4.4.10}$$

Par soucis de simplification, nous noterons par la suite $r^{\mathbf{A}}$ le maximum des TT-rang d'une TT-matrice $\mathbf{A}$.

Nous pouvons réécrire une TT-matrice sous forme de contractions de tenseurs :

$$\mathbf{A} = \mathbf{G}_1(i_1,j_1) \bullet_{\alpha_1} \mathbf{G}_2(i_2,j_2) \bullet_{\alpha_2} \dots \bullet_{\alpha_{d-2}} \mathbf{G}_{d-1}(i_{d-1},j_{d-1}) \bullet_{\alpha_{d-1}} \mathbf{G}_d(i_d,j_d) \tag{4.4.11}$$

Cette TT-matrice est un tensor network selon la figure 4.5 :

FIGURE 4.5 – Tensor network d'une TT-matrice $\mathbf{A}$

Comme pour les TT-vecteurs, nous pouvons exprimer une TT-matrice sous forme de contraction de tenseurs $\mathbf{G}_k$ équivalents aux matrices $\mathbf{G}_k(i_k,j_k)$ pour $i_k \in [\![1,n_k]\!], j_k \in [\![1,m_k]\!]$:

$$\mathbf{A} = \mathbf{G}_1 \bullet_{\alpha_1} \mathbf{G}_2 \bullet_{\alpha_2} \dots \bullet_{\alpha_{d-2}} \mathbf{G}_{d-1} \bullet_{\alpha_{d-1}} \mathbf{G}_d \tag{4.4.12}$$

soit, en développant

$$\mathbf{A}[i_1,\dots,i_d;j_1,\dots,j_d] = \sum_{\alpha_1=1}^{r_1} \dots \sum_{\alpha_{d-1}=1}^{r_{d-1}} \mathbf{G}_1(i_1,j_1,\alpha_1)\mathbf{G}_2(\alpha_1,i_2,j_2,\alpha_2)\dots\mathbf{G}_d(\alpha_{d-1},i_d,j_d) \tag{4.4.13}$$

Ainsi, $\mathbf{A}$ peut être représentée selon un tenseur network

$\forall [\![1,n]\!], \quad \mathbf{G}_k \in \mathbb{R}^{r_{k-1}\times n_k \times m_k \times r_k}$ sont appelés "cores" de la TT-matrice $\mathbf{A}$. $\mathbf{A}$ peut s'exprimer sous forme d'un tenseur d'ordre n.

FIGURE 4.6 – Tensor network d'une TT-matrice $\mathbf{A}$ (format avec contractions)

Remarque importante : ces deux formalismes, un coefficient comme produit de matrices (schéma 4.5) ou une contraction de tenseurs d'ordre 4 le long d'un Tensor Network (schéma 4.6) sont équivalents. Nous poursuivons le développement avec des contractions de tenseurs d'ordre 4 étant donné que c'est le formalisme à la base de plusieurs Toolbox pour l'approche TT comme la TT-toolbox [Ose11] où T3F [NIK+20]. L'article [DH19] présente les formats des cores des TT-matrices dans librairie T3F qui est similaire à celui dans la TT-toolbox.

4.4.3 Opérations élémentaires

Dans la section 4.3, nous avons expliqué le TT-format et dans la section 4.4.2, nous avons présenté les objets TT-vecteurs et TT-matrices qui permettent d'écrire de vecteurs et les matrices dans le format TT. Nous expliquons dans cette section comment se font les opérations élémentaires (additions 4.4.3.1, produits TT-matrice/TT-vecteur 4.4.3.2 et produits de TT-matrices 4.4.3.3) dans ce format. Nous présentons aussi des propriétés sur les TT-rang des objets.

On note pour cette section $\mathbf{X}$ un TT-vecteur, $\mathbf{A}$ et $\mathbf{B}$ deux TT-matrices selon :

$$\begin{aligned}
\mathbf{X}[j_1, j_2, \ldots, j_d] &= \mathbf{G}_1^{\mathbf{X}}(j_1).\mathbf{G}_2^{\mathbf{X}}(j_2).\ldots.\mathbf{G}_d^{\mathbf{X}}(j_d) \\
\mathbf{A}[j_1, i_2, \ldots, i_d; j_1, j_2, \ldots, j_d] &= \mathbf{G}_1^{\mathbf{A}}(i_1, j_1).\mathbf{G}_2^{\mathbf{A}}(i_2, j_2).\ldots.\mathbf{G}_d^{\mathbf{A}}(i_d, j_d) \\
\mathbf{B}[i_1, i_2, \ldots, i_d; j_1, j_2, \ldots, j_d] &= \mathbf{G}_1^{\mathbf{B}}(i_1, j_1).\mathbf{G}_2^{\mathbf{B}}(i_2, j_2).\ldots.\mathbf{G}_d^{\mathbf{B}}(i_d, j_d)
\end{aligned}$$

4.4.3.1 Sommes de TT-matrices

Soit $\mathbf{D} = \mathbf{A} + \mathbf{B}$. Alors, $\mathbf{D}$ s'exprime :

$$\mathbf{D}[i_1, i_2, \ldots, i_d; j_1, j_2, \ldots, j_d] = \mathbf{G}_1^{\mathbf{D}}(i_1, j_1).\mathbf{G}_2^{\mathbf{D}}(i_2, j_2).\ldots.\mathbf{G}_d^{\mathbf{D}}(i_d, j_d)$$

avec :

$$\begin{cases}
\mathbf{G}_1^{\mathbf{D}}(i_1, j_1) = \begin{pmatrix} \mathbf{G}_1^{\mathbf{A}}(i_1, j_1) & \mathbf{G}_1^{\mathbf{B}}(i_1, j_1) \end{pmatrix} \\
\mathbf{G}_d^{\mathbf{D}}(i_d, j_d) = \begin{pmatrix} \mathbf{G}_d^{\mathbf{A}}(i_d, j_d) \\ \mathbf{G}_d^{\mathbf{B}}(i_d, j_d) \end{pmatrix} \\
\forall k \in [\![2, d-1]\!], \quad \mathbf{G}_k^{\mathbf{D}}(i_k, j_k) = \begin{pmatrix} \mathbf{G}_k^{\mathbf{A}}(i_k, j_k) & 0 \\ 0 & \mathbf{G}_k^{\mathbf{B}}(i_k, j_k) \end{pmatrix}
\end{cases} \tag{4.4.14}$$

Les tailles des cores de $\mathbf{D}$ sont :

$$\begin{cases}
k = 1 & \Rightarrow \quad \mathbf{G}_1^{\mathbf{C}}(i_1, j_1) \in \mathbb{R}^{(1 \times (r_1^{\mathbf{A}} + r_1^{\mathbf{B}}))} \\
1 < k < d & \Rightarrow \quad \mathbf{G}_k^{\mathbf{C}}(i_k, j_k) \in \mathbb{R}^{(r_{k-1}^{\mathbf{A}} + r_{k-1}^{\mathbf{B}}) \times (r_k^{\mathbf{A}} + r_k^{\mathbf{B}})} \\
k = d & \Rightarrow \quad \mathbf{G}_d^{\mathbf{C}}(i_d, j_d) \in \mathbb{R}^{((r_{d-1}^{\mathbf{A}} + r_{d-1}^{\mathbf{B}}) \times 1)}
\end{cases} \tag{4.4.15}$$

Proposition 1. *Rang de la décomposition TT d'une somme de TT-matrice Soient $r^{\mathbf{A}}$ et $r^{\mathbf{B}}$ les rangs maximaux de la décomposition TT de $\mathbf{A}$ et $\mathbf{B}$. Alors, le rang maximal de $\mathbf{D}$ au format TT vérifie :*

$$r^{\mathbf{D}} \leq r^{\mathbf{A}} + r^{\mathbf{B}} \tag{4.4.16}$$

4.4.3.2 Produits de TT-matrices TT-vecteurs

Soit $\mathbf{Y} = \mathbf{A}.\mathbf{X}$. $\mathbf{Y}$ est un TT-vecteur qui s'écrit sous forme de tensor network :

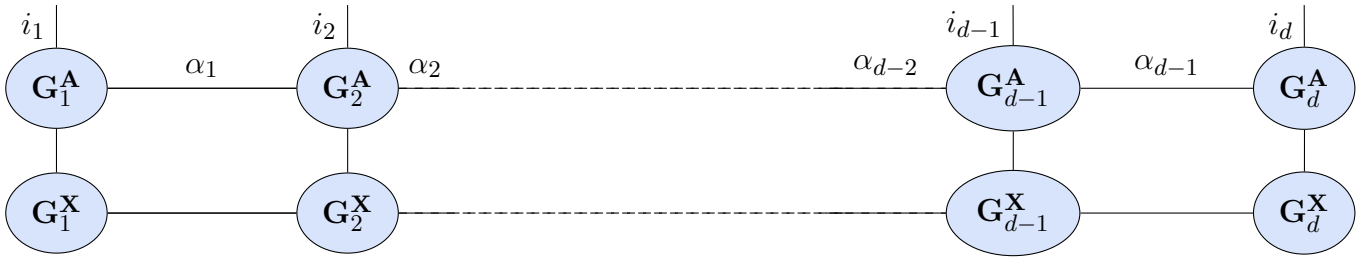


FIGURE 4.7 – Tensor network de la TT-matrice $\mathbf{Y}$ résultante du produit de TT-matrices TT-vecteurs (format avec contraction)

Pour le voir, nous développons les contractions des produit des cores de $\mathbf{A}$ et $\mathbf{X}$ pour obtenir :

$$\begin{aligned}
&\mathbf{Y}[i_1, i_2, \ldots, i_d] = \\
&\sum_{j_1, j_2 \ldots j_d} \left(\sum_{\alpha_1} \ldots \sum_{\alpha_{d-1}} \mathbf{G}_1^{\mathbf{A}}(i_1, j_1)[1, \alpha_1] \mathbf{G}_2^{\mathbf{A}}(i_2, j_2)[\alpha_1, \alpha_2] \ldots \mathbf{G}_d^{\mathbf{A}}(i_d, j_d)[\alpha_{d-1}, 1] \right) \\
&\left(\sum_{\alpha_1} \ldots \sum_{\beta_{d-1}} \mathbf{G}_1^{\mathbf{X}}(j_1)[1, \beta_1] \mathbf{G}_2^{\mathbf{X}}(j_2)[\beta_1, \beta_2] \ldots \mathbf{G}_d^{\mathbf{X}}(j_d)[\beta_{d-1}, 1] \right) \\
&= \sum_{j_1, j_2 \ldots j_d} \sum_{\alpha_1, \alpha_2, \ldots, \alpha_{d-1}} \sum_{\beta_1, \beta_{d-1}} \left(\mathbf{G}_1^{\mathbf{A}}(i_1, j_1)[1, \alpha_1] \mathbf{G}_1^{\mathbf{X}}(j_1)) [1, \beta_1] \right). \\
&\left(\mathbf{G}_2^{\mathbf{A}}(i_2, j_2)[\alpha_1, \alpha_2] \mathbf{G}_2^{\mathbf{X}}(j_2)[\beta_1, \beta_2] \right) \\
&\ldots\ldots \left(\mathbf{G}_d^{\mathbf{A}}(i_d, j_d)[\alpha_{d-1}, 1] \mathbf{G}_d^{\mathbf{X}}(j_d)[\beta_{d-1}, 1] \right)
\end{aligned} \tag{4.4.17}$$

Alors, le TT-vecteur $\mathbf{Y}$ résultant s'exprime :

$$\mathbf{Y}[i_1, i_2, \ldots, i_d] = \mathbf{G}_1^{\mathbf{Y}}(i_1).\mathbf{G}_2^{\mathbf{Y}}(i_2).\ldots.\mathbf{G}_d^{\mathbf{Y}}(i_d)$$

Avec :

$$\begin{cases}
k = 1 & \Rightarrow \quad \mathbf{G}_1^{\mathbf{Y}}(i_1)[\alpha_1, \beta_1] = \sum_{j_1} \mathbf{G}^{\mathbf{A}}(i_1, j_1)[1, \alpha_1] \mathbf{G}^{\mathbf{X}}(j_1)[\beta_1, 1] \\
1 < k < d & \Rightarrow \quad \mathbf{G}_k^{\mathbf{Y}}(i_k)[\alpha_{k-1}, \alpha_k, \beta_{k-1}, \beta_k] = \sum_{j_k} \mathbf{G}^{\mathbf{A}}(i_k, j_k)[\alpha_{k-1}, \alpha_k] \mathbf{G}^{\mathbf{X}}(j_k)[\beta_{k-1}, \beta_k] \\
k = d & \Rightarrow \quad \mathbf{G}_d^{\mathbf{Y}}(i_d)[\alpha_d, \beta_d] = \sum_{j_d} \mathbf{G}_d^{\mathbf{A}}(i_d, j_d)[\alpha_{d-1}, 1] \mathbf{G}_d^{\mathbf{X}}(j_d)[\beta_{d-1}, 1]
\end{cases} \tag{4.4.18}$$

Nous notons $r_k^{\mathbf{A}}$, les rangs des cores de $\mathbf{A}$ et $r_k^{\mathbf{X}}$, les rangs des cores de $\mathbf{X}$.

$$\begin{cases}
k = 1 & \Rightarrow \quad \mathbf{G}_1^{\mathbf{Y}}(i_1) \in \mathbb{R}^{(1 \times (r_1^{\mathbf{A}} \times r_1^{\mathbf{X}}))} \\
1 < k < d & \Rightarrow \quad \mathbf{G}_k^{\mathbf{Y}}(i_k) \in \mathbb{R}^{(r_{k-1}^{\mathbf{A}} \times r_{k-1}^{\mathbf{X}}) \times (r_k^{\mathbf{A}} \times r_k^{\mathbf{X}})} \\
k = d & \Rightarrow \quad \mathbf{G}_d^{\mathbf{Y}}(i_d) \in \mathbb{R}^{((r_{d-1}^{\mathbf{A}} \times r_{d-1}^{\mathbf{X}}) \times 1)}
\end{cases} \tag{4.4.19}$$

Proposition 2. *Á partir de la formulation de* $\mathbf{Y}$*, nous pouvons déduire :*

- $\mathbf{Y}$ *est un TT-vecteur (tenseur d'ordre d) et son TT-rang maximal est borné par :*

$$r^{\mathbf{Y}} \leq r^{\mathbf{A}} \times r^{\mathbf{X}} \tag{4.4.20}$$

- *Si* $\mathbf{A}$ *est une TT-vecteur, alors* $\mathbf{Y} = \mathbf{A}.\mathbf{X}$ *est une TT-matrice qui peut être transformée en scalaire par une contraction sur ses cores.*

4.4.3.3 Produits de TT-matrices

Soit,

$$\mathbf{C} = \mathbf{A}.\mathbf{B}$$

La TT-matrice $\mathbf{C}$ est un tensor network qui s'exprime selon la figure 4.8 :

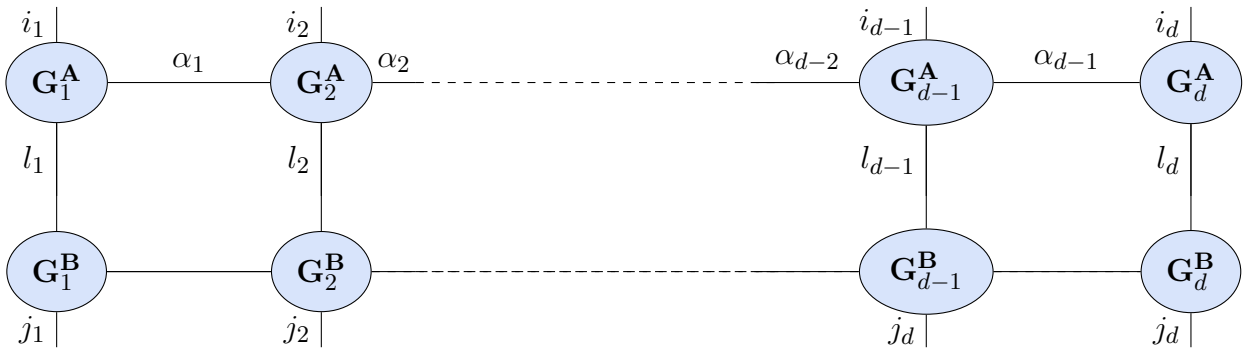


FIGURE 4.8 – Tensor network de la TT-matrice $\mathbf{C}$ résultante d'un produit TT-matrices (format avec contraction)

En développant les contractions des produit des cores de $\mathbf{A}$ et $\mathbf{B}$, nous obtenons :

$$\begin{aligned}
&\mathbf{C}[i_1, i_2, \ldots, i_d; j_1, j_2, \ldots, j_d] = \\
&\sum_{l_1, l_2 \ldots l_d} \left(\sum_{\alpha_1} \cdots \sum_{\alpha_{d-1}} \mathbf{G}_1^{\mathbf{A}}(i_1, l_1)[1, \alpha_1] \mathbf{G}_2^{\mathbf{A}}(i_2, l_2)[\alpha_1, \alpha_2] \ldots \mathbf{G}_d^{\mathbf{A}}(i_d, l_d)[\alpha_{d-1}, 1] \right) \\
&\left(\sum_{\alpha_1} \cdots \sum_{\beta_{d-1}} \mathbf{G}_1^{\mathbf{B}}(l_1, j_1)[1, \beta_1] \mathbf{G}_2^{\mathbf{B}}(l_2, j_2)[\beta_1, \beta_2] \ldots \mathbf{G}_d^{\mathbf{B}}(l_d, j_d)[\beta_{d-1}, 1] \right) \\
&= \sum_{l_1, l_2 \ldots l_d} \sum_{\alpha_1, \alpha_2, \ldots, \alpha_{d-1}} \sum_{\beta_1, \beta_{d-1}} \left(\mathbf{G}_1^{\mathbf{A}}(i_1, l_1)[1, \alpha_1] \mathbf{G}_1^{\mathbf{B}}(l_1, j_1)[1, \beta_1] \right) \\
&\qquad \left(\mathbf{G}_2^{\mathbf{A}}(i_2, l_2)[\alpha_1, \alpha_2] \mathbf{G}_2^{\mathbf{B}}(l_2, j_2)[\beta_1, \beta_2] \right) \\
&\qquad \ldots\ldots \left(\mathbf{G}_d^{\mathbf{A}}(i_d, l_d)[\alpha_{d-1}, 1] \mathbf{G}_d^{\mathbf{B}}(i_d, j_d)[\beta_{d-1}, 1] \right)
\end{aligned} \tag{4.4.21}$$

Alors, la TT-matrice $\mathbf{C}$ s'exprime :

$$\mathbf{C}[i_1, i_2, \ldots, i_d; j_1, j_2, \ldots, j_d] = \mathbf{G}_1^{\mathbf{C}}(i_1, j_1).\mathbf{G}_2^{\mathbf{C}}(i_2, j_2).\ldots.\mathbf{G}_d^{\mathbf{C}}(i_d, j_d)$$

Soit $\forall k \in [\![1, d]\!]$, nous notons $r_k^{\mathbf{A}}$, les rangs des cores de $\mathbf{A}$ et $r_k^{\mathbf{B}}$, les rangs des cores de $\mathbf{B}$. Alors les cores de $\mathbf{C}$ s'expriment :

$$\begin{cases} k=1 & \Rightarrow \quad \mathbf{G}_1^{\mathbf{C}}(i_1,j_1)[\alpha_1,\beta_1] = \sum_{l_1} \mathbf{G}^{\mathbf{A}}(i_1,l_1)[1,\alpha_1]\mathbf{G}^{\mathbf{B}}(l_1,j_1)[\beta_1,1] \\ 1<k<d & \Rightarrow \quad \mathbf{G}_k^{\mathbf{C}}(i_k,j_k)[\alpha_{k-1},\alpha_k,\beta_{k-1},\beta_k] = \sum_{l_k} \mathbf{G}^{\mathbf{A}}(i_k,l_k)[\alpha_{k-1},\alpha_k]\mathbf{G}^{\mathbf{B}}(l_k,j_k)[\beta_{k-1},\beta_k] \\ k=d & \Rightarrow \quad \mathbf{G}_d^{\mathbf{C}}(i_d,j_d)[\alpha_d,\beta_d] = \sum_{l_d} \mathbf{G}_d^{\mathbf{A}}(i_d,l_d)[\alpha_{d-1},1]\mathbf{G}_d^{\mathbf{B}}(i_d,j_d)[\beta_{d-1},1] \end{cases} \tag{4.4.22}$$

Nous notons $r_k^{\mathbf{A}}$, les rangs des cores de $\mathbf{A}$ et $r_k^{\mathbf{B}}$, les rangs des cores de $\mathbf{B}$. et

$$\begin{cases} k=1 & \Rightarrow \quad \mathbf{G}_1^{\mathbf{C}}(i_1,j_1) \in \mathbb{R}^{(1\times(r_1^{\mathbf{A}}\times r_1^{\mathbf{B}}))} \\ 1<k<d & \Rightarrow \quad \mathbf{G}_k^{\mathbf{C}}(i_k,j_k) \in \mathbb{R}^{(r_{k-1}^{\mathbf{A}}\times r_{k-1}^{\mathbf{B}})\times(r_k^{\mathbf{A}}\times r_k^{\mathbf{B}})} \\ k=d & \Rightarrow \quad \mathbf{G}_d^{\mathbf{C}}(i_d,j_d) \in \mathbb{R}^{((r_{d-1}^{\mathbf{A}}\times r_{d-1}^{\mathbf{B}})\times 1)} \end{cases} \tag{4.4.23}$$

Proposition 3. *Á partir de la formulation de* $\mathbf{C}$*, nous pouvons déduire que* $\mathbf{C}$ *est une TT-matrice (tenseur d'ordre d) et le rang maximal de sa décomposition TT est borné par :*

$$r^{\mathbf{C}} \leq r^{\mathbf{A}} \times r^{\mathbf{B}} \tag{4.4.24}$$

Remarque : D'autres opérations sur les TT-matrices sont possibles : l'article [LC18] en présente certaines comme la contraction totale, le produits d'Hadamard ...

4.4.4 Compression des TT-matrices : utilisation du rounding

Nous avons vu dans la section précédente que le produit de deux TT-matrices augmente les rangs maximaux des cores de la TT-matrice résultante, selon $r^{\mathbf{C}} \leq r^{\mathbf{A}} \times r^{\mathbf{B}}$ si $\mathbf{C} = \mathbf{A}.\mathbf{B}$. Une manière de réduire ces rangs est de faire une compression de la TT-matrice résultante en l'approchant au mieux par une TT-matrice de rang plus faible. Cette démarche se réalise selon la procédure dite de "rounding", proposé par Oseledets [Ose11] pour approcher un tenseur en format TT par un tenseur en format TT de rang plus faible, et étendue ensuite à l'approximation de TT-vecteurs et aux TT-matrices. Nous utiliserons dans ce document essentiellement le rounding sur des TT-matrices et TT-vecteurs.

Il s'agit simplement une TT-SVD d'un tenseur qui est déjà dans le format format TT. Les détails de sa mise en oeuvre sont cependant très techniques, et ne seront pas repris intégralement ici. Si

$$\mathbf{A}[i_1,i_2,\ldots,i_d;j_1,j_2,\ldots,j_d] = \mathbf{G}_1(i_1,j_1).\mathbf{G}_2(i_2,j_2).\ldots.\mathbf{G}_d(i_d,j_d)$$

l'algorithme du rounding prend en entrée une TT-matrice $\mathbf{A}$ et une erreur ϵ ou un rang prescrit r. Comme dans la TT-toolbox que nous avons utilisée, la précision prescrite est favorisée, nous expliquons la version de cet algorithme avec précision prescrite. L'algorithme retourne alors $\overline{\mathbf{A}}$ en format TT de rang le plus faible tel que

$$||\mathbf{A} - \overline{\mathbf{A}}|| \leq \epsilon ||\mathbf{A}||$$

Le rounding se fait en deux phases :

- **Phase d'orthogonalisation :**
  Elle consiste à parcourir séquentiellement les d cores de $\mathbf{A}$ de la droite vers la gauche (c'est-à-dire du $d-$ième core au premier) et réaliser une décomposition LQ de la matrice de chaque core.

- **Phase de compressions :**
  Consiste à parcourir séquentiellement les d cores orthogonaux de la gauche vers la droite en appliquant des décompositions SVD avec une erreur ϵ prescrite ou un rang r prescrit de chaque core.

Cet algorithme est décrit de manière détaillée dans l'article d'Oseledets [Ose11] dans le cadre des Tenseurs au format TT et exprimé pour les TT matrices dans l'article [DBNW20] auxquels nous renvoyons pour les détails. Le rounding comme l'approximation de rang faible de matrices par la SVD peut se faire à précision prescrite ϵ ou à rang prescrit r. Ces deux possibilités sont offertes par les toolbox que nous avons utilisées : la TT-toolbox [Ose11] ou T3F [NIK+20].

4.5 Intuition de l'approche TT appliquée au calcul des marginales dans un modèle graphiques

4.5.1 Modèles graphiques

Soit $n, m \in \mathbb{N}$,

Un graphe $G = (V, E)$ est composé d'un ensemble de nœuds V et d'un ensemble d'arêtes E :

- $V = (1, ..., n)$ représente l'ensemble fini des nœuds (vertices) de G.
- $E \subset V \times V$ représente un ensemble fini d'arêtes (edges).
  Chacune d'entre elles représente la connexion entre deux nœuds.

Modèle graphique [Bis06] Soit $G = (V, E)$ un graphe où l'état de chaque nœud $i \in V$ est une variable aléatoire notée Z_i. Nous nous intéressons à la loi jointe $\mathbb{P}(Z_1, \ldots, Z_n)$. Nous considérons un ensemble $\mathcal{A}$ de m parties de V

$$\mathcal{A} = (A_1, \ldots, A_m), \qquad A_\ell \subset V$$

Nous notons Z_{A_ℓ} ou plus simplement Z^ℓ l'ensemble

$$Z_{A_\ell} = (Z_{i_1}, \ldots, Z_{i_\ell})$$

si

$$A_\ell = (i_1, \ldots, i_{m_\ell}), \qquad m_\ell = |A_\ell|.$$

Alors, $Z = (Z_1, \ldots, Z_n)$ suit un modèle graphique s'il existe des fonctions réelles ψ_ℓ pour chaque partie A_ℓ telles que

$$\mathbb{P}(z_1, \ldots, z_n) \propto \prod_{\ell=1}^{m} \psi_\ell(z_{A_\ell}) \tag{4.5.1}$$

Les fonctions ψ_{A_ℓ} sont appelées les facteurs du modèle graphique. Nous appelons hyperlien un élément $A_\ell \subset V$, et la paire $H = (V, \mathcal{A})$ est appelée un hypergraphe.

Nous introduisons alors la constante de normalisation

$$W = \sum_z \left(\prod_{\ell=1}^{m} \psi_\ell(z_{A_\ell}) \right)$$

Nous avons alors,

$$\mathbb{P}(z_1, \ldots, z_n) = \frac{1}{W} \prod_{\ell=1}^{m} \psi_\ell(z_{A_\ell}) \tag{4.5.2}$$

4.5.2 Intuition de l'approche TT

Commençons par présenter l'idée qui motive l'approche Tenseur Train et par la comparer avec le champ moyen. Tout d'abord ces deux approches sont des approximations avec séparation de variables dans un modèle graphique, ce qui facilite la marginalisation. Utilisons un modèle graphique simple comme présenté dans la figure 5.1 comme support :

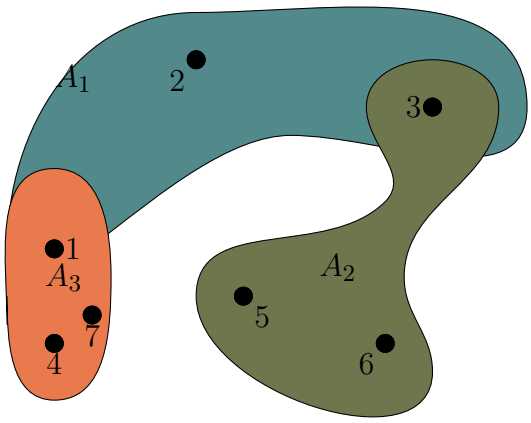


FIGURE 4.9 – Exemple de modèle graphique

Soit n le nombre d'individus dans ce modèle, Q le nombre d'états de chaque individu et m le nombre d'hyperliens. Soit Ψ une fonction qui exprime le modèle graphique présenté dans la figure 4.9,

$$\Psi[z_1, \ldots, z_7] = \prod_{\ell=1,\ldots,3} \psi_{A_\ell}(z_{A_\ell})$$

Les différents hyperliens de ce modèle sont : $A_1 = (z_1, z_2, z_3), A_2 = (z_3, z_5, z_6)$ et $A_3 = (z_1, z_4, z_7)$. On en déduit,

$$\Psi[z_1, \ldots, z_7] = \psi_{A_1}(z_1, z_2, z_3)\psi_{A_2}(z_3, z_5, z_6)\psi_{A_3}(z_1, z_4, z_7)$$

- À partir de cette fonction, les marginales s'écrivent comme :

$$\forall i \in [\![1, 7]\!], \qquad p_i(z_i) = \sum_{z_1, z_2, \ldots, z_{i-1}, z_{i+1}, \ldots, z_7} \Psi[z_1, \ldots, z_7]$$

  Le calcul de la constante de normalisation se fait ainsi :

$$W = \sum_{z_1, \ldots, z_7} \Psi(z_1, z_2, \ldots, z_7)$$

  On développe cette formule et on obtient :

$$W = \sum_{z_1=1}^{Q} \sum_{z_2=1}^{Q} \sum_{z_3=1}^{Q} \sum_{z_4=1}^{Q} \sum_{z_5=1}^{Q} \sum_{z_6=1}^{Q} \sum_{z_7=1}^{Q} \psi_{A_1}(z_1, z_2, z_3)\psi_{A_2}(z_3, z_5, z_6)\psi_{A_3}(z_1, z_4, z_7)$$

  On réorganise cette formule :

$$W = \sum_{z_3=1}^{Q} \sum_{z_5=1}^{Q} \sum_{z_6=1}^{Q} \psi_{A_3}(z_3, z_5, z_6) \left[\sum_{z_1=1}^{Q} \sum_{z_2=1}^{Q} \psi_{A_1}(z_1, z_2, z_3) \left(\sum_{z_4=1}^{Q} \sum_{z_7=1}^{Q} \psi_{A_3}(z_1, z_4, z_7) \right) \right] \tag{4.5.3}$$

  Nous constatons que ce calcul nécessite $3Q^3$ sommes. Pour les marginales unaires, le calcul nécessite $2Q^3 + Q^2$ sommes.
  Une solution pour y remédier est d'approcher ψ par une fonction avec séparation des variables : d'où l'approche champ moyen.

- **Procédé pour la séparation des variables :**
  Soit,
  $$\psi(z_1, z_2, \ldots, z_n) = f_1(z_1) f_2(z_2) \ldots f_n(z_n)$$
  alors, par exemple,
  $$p_1(z_1) = f_1(z_1) \left[\sum_{z_2} f_2(z_2) \right] \ldots \left[\sum_{z_n} f_n(z_n) \right]$$
  Cette formulation nécessite $(n-1)Q$ sommes.

Grâce à la séparation des variables, nous réduisons considérablement le nombre de sommes.

- **Le champ moyen :** consiste en une approximation de la loi jointe par une distribution qui est le produit des facteurs unaires pour laquelle les marginales sont plus simples à calculer.
  En effet,
  $$\text{si} \qquad x_{ij} = a_i b_j$$
  $$\text{alors,} \qquad \underbrace{\textstyle\sum_{i,j} x_{ij}}_{n^2\text{sommes}} = \textstyle\sum_{i,j} a_i b_j = \underbrace{(\textstyle\sum_i a_i)(\textstyle\sum_j b_j)}_{2n\text{ sommes}}$$
  Nous recherchons une approximation optimale au sens de la divergence Kullback-Leibler.

- **Tenseur Train** : l'approche Tenseur Train est une généralisation de l'approche champ moyen.
  On se donne une matrice $X \in \mathbb{R}^{n \times n}$ d'éléments $x_{ij} \in \mathbb{R}$, alors $\mathbf{w} = \sum_{i,j} x_{ij}$ nécessite n^2 sommes. Nous supposons maintenant qu'il existe deux familles de vecteurs $(\mathbf{a}_i)_i, (\mathbf{b}_j)_j$ telles que :
  $$\forall i, j \qquad x_{ij} = \mathbf{a}_i \mathbf{b}_j \qquad \begin{cases} \mathbf{a}_i & \in \mathbb{R}^{1 \times n} \\ \mathbf{b}_j & \in \mathbb{R}^{n \times 1} \end{cases} \tag{4.5.4}$$
  On considère que $\mathbf{a}_i$ est une matrice $1 \times n$ (horizontale) et $\mathbf{b}_j$ est une matrice $n \times 1$ (verticale), donc x_{ij} est un produit de matrices.Nous pouvons donc écrire :
  $$\sum_{i,j} x_{ij} = \sum_{i,j} \mathbf{a}_i \mathbf{b}_j = (\sum_i \mathbf{a}_i) \cdot (\sum_j \mathbf{b}_j)$$
  avec $2n$ sommes de matrices et un produit matriciel.
  L'approche TT pour le calcul de $w = \sum_{i,j} x_{ij}$pour X (matrice quelconque) consiste à trouver la matrice X' de termes $x'_{ij} = \mathbf{a}_i \cdot \mathbf{b}_j$ la plus proche de X au sens de la norme de Frobenius et calculer $w(X')$.

Dans l'approche champ moyen, la loi avec variables séparées choisie est la plus proche de ψ au sens de la divergence Kullback-Leibler alors que pour l'approche TT, la distance est liée à la norme de Frobenius. De façon plus générale, nous approchons la loi jointe d'un modèle graphique par le produit matriciel suivant :
$$\widetilde{\psi}[z_1, \ldots, z_n] = u(z_1) \cdot G_2(z_2) \cdot \ldots \cdot G_{n-1}(z_{n-1}) \cdot v(z_n) \qquad \begin{cases} u(z_1) & \in \mathbb{R}^{1 \times r} \\ G_i(z_i) & \in \mathbb{R}^{r \times r} \\ v(z_n) & \in \mathbb{R}^{r \times 1} \end{cases} \tag{4.5.5}$$
qui est une forme de séparation des variables (chaque coefficient $\psi'[z_1, \ldots, z_n]$ est un produit de matrices qui dépendent chacune d'une variable uniquement).

4.6 Résumé de l'approche TT pour calculer les marginales d'un modèle graphique

On prend le cas d'un modèle graphique avec $n \in \mathbb{N}$ nœuds, $Q \in \mathbb{N}$ états et $\Lambda = [\![1, Q]\!]$ l'espace des états (c'est-à-dire que $\forall i \in \mathbb{N}, z_i \in \Lambda$). Soit z l'ensemble des états du modèle. Sa loi jointe s'exprime :

$$\begin{aligned} \psi : \Lambda^n &\xrightarrow{\psi} \mathbb{R} \\ (z_1, \ldots, z_n) &\longrightarrow \psi(z) \end{aligned} \tag{4.6.1}$$

Nous notons z^ℓ le tuple des états des nœuds intervenant dans l'hyperlien A_ℓ et $\psi_{A_\ell}(z^\ell)$ les facteurs associés à cet hyperlien. Alors, la loi jointe peut s'exprimer :

$$\psi(z_1, \ldots, z_n) = \prod_{\ell=1}^{m} \psi_{A_\ell}(z^\ell) \tag{4.6.2}$$

Pour alléger nos formules, nous simplifions certaines notations : ψ_{A_ℓ} devient ψ^ℓ et $z = (z_1, \ldots, z_n)$. Ainsi la formule (4.6.2) s'exprime :

$$\psi(z) = \prod_{\ell} \psi^\ell(z^\ell) \tag{4.6.3}$$

Ensuite, comme $\forall \ell \in [\![1, m]\!], \psi^\ell(z^\ell) \in \mathbb{R}$,nous utilisons la propriété P_2 pour réécrire la formule (4.6.3) avec un produit de Kronecker.

$$\psi(z) = \psi^1(z^1) \otimes_k \psi^2(z^2) \otimes_k \ldots \otimes_k \psi^m(z^m) \tag{4.6.4}$$

Cela nous permettra d'utiliser la propriété du produit mixte (P_1 "mixed-product poperty") plus tard.

- **Étape 1 : approximation des facteurs ψ_ℓ, $\forall \ell = 1, \ldots, m$ au format TT .** Cette étape se décompose en deux parties :

  Soit $i_1, \ldots, i_{m_\ell} \in [\![1, n]\!]^{m_\ell}$, $A_\ell = \{i_1, \ldots, i_{m_\ell}\}$ les noeuds intervenant dans (A_ℓ) de cardinal $m_\ell = \text{card}(A_\ell)$. Nous notons z_{i_f} l'état du noeud i_f et z^ℓ les états des noeuds intervenant dans (A_ℓ).

  Mise des facteurs sous format tensoriel. Pour chaque hyperlien $\ell = 1, \ldots, m$, nous associons un tenseur indexé par z^ℓ : ψ^ℓ d'ordre m_ℓ, de modes $i_1, \ldots, i_{m_\ell}$ et de dimensions Q^{m_ℓ}

  TT-SVD. Nous faisons une TT-SVD des tenseurs associés aux facteurs ψ^ℓ : l'approximation TT du facteur ℓ se décompose selon :

  $$\psi^\ell(z_{i_1}, \ldots, z_{i_{m_\ell}}) = G^\ell_{i_1}(z_{i_1}) \cdot G^\ell_{i_2}(z_{i_2}) \cdot \ldots \cdot G^\ell_{i_{m_\ell}}(z_{i_{m_\ell}}) \tag{4.6.5}$$

  $G^\ell_{i_1}(z_{i_1}), \ldots G^\ell_{i_{m_\ell}}(z_{i_{m_\ell}})$ sont appelés les cores de l'approximation TT des facteurs. Ils sont de tailles : $G^\ell_{i_1}(z_{i_1}) \in \mathbb{R}^{1 \times r_1}$, $G^\ell_{i_{m_\ell}}(z_{i_{m_\ell}}) \in \mathbb{R}^{r_{i_{m_\ell}} \times 1}$ et $\forall k \in [\![1, m_\ell]\!]$, $G^\ell_{i_k}(z_{i_k}) \in \mathbb{R}^{r_{k-1} \times r_k}$ où r est le rang maximal de au format TT du facteur comme nous l'avons expliqué dans l'état de l'art sur les tenseurs.
  Nous voyons apparaître la séparation des variables via le format TT pour chaque facteur. La suite de la démarche est de construire une séparation des variables de type format TT pour la loi jointe (produit de tous les facteurs). cela se fait via un "trick" proposé par Novikov & al. (2014) : l'ajout de variables non essentielles

- **Étape 2 : Ajout des dimensions non essentielles .** Deux facteurs ℓ_1 et ℓ_2 différent selon les variables dans A_ℓ. Nous présenterons un exemple simple pour $n = 4$ et $m = 2$:

$$\psi(z) = \psi^1(z_1, z_2, z_3)\psi^2(z_2, z_3, z_4)$$

Nous avons :

$$\psi^1(z_1, z_2, z_3) = G_1^1(z_1)G_1^2(z_2)G_1^3(z_3)$$
$$\psi^2(z_2, z_3, z_4) = G_2^2(z_2)G_2^3(z_3)G_2^4(z_4)$$

Nous rajoutons les variables dites non essentielles par Novikov [NROV14] de telle façon que les nouveaux facteurs dépendent des mêmes variables. Nous écrivons :

$$\overline{\psi}^1(z_1, z_2, z_3) = G_1^1(z_1)G_1^2(z_2)G_1^3(z_3)G_1^4(z_4)$$

avec $G_1^4(z_4) = 1 \in \mathbb{R}$

$$\overline{\psi}^2(z_2, z_3, z_4) = G_2^1(z_1)G_2^2(z_2)G_2^3(z_3)G_2^4(z_4)$$

avec $G_2^1(z_1) = 1 \in \mathbb{R}$

Il est possible de montrer [NROV14] que ce complément est toujours possible, avec des choix $\forall k \in [\![1, n]\!], G_k^\ell(z_k) = 1$ ou $\mathbb{I}_r$ pour les variables non essentielles selon la règle :
Si

$$\psi^\ell(z_{i_1}, \ldots, z_{i_{m_\ell}}) = \prod_{k=1}^{i_{m_\ell}} G_k^\ell(z_k) \tag{4.6.6}$$

Nous l'exprimons en fonction de tous les individus :

$$\overline{\psi^\ell}(z^\ell) = \prod_{k=1}^{n} \varphi_k^\ell(z_k) \tag{4.6.7}$$

Avec

$$\left|\begin{array}{lll} k < i_1 & \Rightarrow \varphi_k^\ell(z_k) = 1 & \in \mathbb{R}^{1\times 1} \\ k = i_1 & \Rightarrow \varphi_k^\ell(z_k) = G_{i_1}^\ell(z_{i_1}) & \in \mathbb{R}^{1\times r_{i_1}} \\ i_1 < k < i_{m_\ell}, k \notin A_\ell & \Rightarrow \varphi_k^\ell(z_k) = \mathbb{I}_{Q,Q} & \in \mathbb{R}^{r_{k-1}\times r_k} \\ i_1 < k < i_{m_\ell}, k \in A_\ell & \Rightarrow \varphi_k^\ell(z_k) = G_k^\ell(z_k) & \in \mathbb{R}^{r_{k-1}\times r_k} \\ k = i_{m_\ell} & \Rightarrow \varphi_k^\ell(z_k) = G_{i_{m_\ell}}^\ell(z_{i_{m_\ell}}) & \in \mathbb{R}^{r_{i_{m_\ell}-1}\times 1} \\ k > i_{m_\ell} & \Rightarrow \varphi_k^\ell(z_k) = 1 & \in \mathbb{R}^{1\times 1} \end{array}\right. \tag{4.6.8}$$

Remarque 4. *$\overline{\psi^\ell}(z^\ell)$ ne dépend que de $z_{i_1}, \ldots, z_{i_{m_\ell}}$ et pas des z_k pour $k \notin A_\ell$ qui sont des variables muettes donc :*

$$\psi^\ell(z_{i_1}, \ldots, z_{i_{m_\ell}}) = \overline{\psi^\ell}(z^\ell)$$

- **Étape 3 : Écriture de $\psi(z)$ avec rajout des dimensions non essentielles.** Cette écriture repose sur la propriété "Mixed-product property". et utilise la formule (4.6.7) afin de réécrire la formule (4.6.4) de cette manière :

$$\psi(z) = \psi_1(z) \otimes_k \psi_2(z) \otimes_k \ldots \otimes_k \psi_m(z)$$

$$\begin{aligned} &= (\varphi_1^1(z_1) \cdot \varphi_2^1(z_2) \cdot \cdots \cdot \varphi_n^1(z_n)) \otimes_k \\ &\quad (\varphi_1^2(z_1) \cdot \varphi_2^2(z_2) \cdot \cdots \cdot \varphi_n^2(z_n)) \otimes_k \\ &\quad \cdots \otimes_k (\varphi_1^m(z_1) \cdot \varphi_2^m(z_2) \cdot \cdots \cdot \varphi_n^m(z_n)) \end{aligned} \tag{4.6.9}$$

Ensuite, nous utilisons la Mixed-product property (P_2) pour exprimer $\psi(z)$:

$$\begin{aligned}\psi(z) = (\varphi_1^1(z_1) \otimes_k \varphi_1^2(z_1) \otimes_k \cdots \otimes_k \varphi_1^m(z_1))\cdot \\ (\varphi_2^1(z_2) \otimes_k \varphi_2^2(z_2) \otimes_k \cdots \otimes_k \varphi_2^m(z_2))\cdot \\ \cdots\cdot(\varphi_n^1(z_n) \otimes_k \varphi_n^2(z_n) \otimes_k \cdots \otimes_k \varphi_n^m(z_n))\end{aligned} \tag{4.6.10}$$

- **Étape 4 : Séparation des variables**. Nous notons :

$$A_i[z_i] = \otimes_{k\ell=1}^m \varphi_i^\ell(z_i) \tag{4.6.11}$$

À partir de la formule (4.6.10) et (4.6.11) nous avons :

$$\psi(z) = \prod_{i=1}^n A_i[z_i] \tag{4.6.12}$$

Cette écriture dans un format de produit matriciel est une écriture de $\psi(z)$ avec une séparation des variables z_i. C'est l'écriture en format TT du tenseur associé à la loi jointe $\psi(z)$.

- **Étape 5 : Marginalisation.** Grâce à la séparation des variables,nous pouvons calculer facilement différentes marginalisations comme la constante de normalisation, des marginales unaires et binaires qu'on présente ci-dessous :

  - **Constante de normalisation :**

$$W = \sum_z \psi(z)$$

On utilise la propriété de distributivité du produit sur les sommes pour développer notre calcul et nous obtenons :

$$W = \sum_{z_1} \cdots \sum_{z_n} \left(\prod_{i=1}^n A_i[z_i]\right) = \sum_{z_1} \cdots \sum_{z_n} A_1[z_1] \cdots A_2[z_2] \cdots \cdots \cdots A_n[z_n]$$

Ainsi, la constante de normalisation s'exprime :

$$W = \left(\sum_{z_1} A_1[z_1]\right) \cdot \left(\sum_{z_2} A_2[z_2]\right) \cdots \cdot \left(\sum_{z_n} A_n[z_n]\right) \tag{4.6.13}$$

On pose :

$$B_i = \sum_{z_i} A_i[z_i]$$

On peut transformer la formule (4.6.13) et exprimer la constante de normalisation de cette manière :

$$W = B_1 B_2 \cdots B_n \tag{4.6.14}$$

  - **Marginales unaires :**

$$\forall i \in [\![1, n]\!], \qquad p_i(z_i) = \sum_{z_1, z_2, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n} \psi(z)$$

On redéveloppe cette formule de la même manière (nous notons $\times$ le produit matriciel).

$$p_i(z_i) = \left(\sum_{z_1} A_1[z_1]\right) \cdot \left(\sum_{z_2} A_2[z_2]\right) \cdots \left(\sum_{z_{i-1}} A_{i-1}[z_{i-1}]\right) \cdot A_{i+1}[z_{i+1}] \cdot \left(\sum_{z_{i-1}} A_{i+1}[z_{i+1}]\right) \cdots \left(\sum_{z_n} A_n[z_n]\right) \tag{4.6.15}$$

$$p_i(z_i) = B_1 \ldots B_{i-1} \mathbf{A}_i[z_i] B_{j+1} \ldots B_n \tag{4.6.16}$$

- **Marginales binaires :** De la même manière que pour les marginales unaires, nous faisons le calcul des marginales binaires :

$$\forall i,j \in [\![1,n]\!]^2, \qquad p_{ij}(z_i,z_j) = \sum_{z_1,z_2,\dots,z_{i-1},z_{i+1},\dots,z_{j-1},z_{j+1},\dots z_n} \psi(z)$$

$$p_{i,j}(z_i,z_j) = B_1 \dots B_{i-1}\mathbf{A}_i[z_i]B_{i+1}\dots B_{j-1}\mathbf{A}_j[z_j]B_{j+1}\dots B_n \tag{4.6.17}$$

4.7 Conclusion

Dans ce chapitre, nous avons présenté les concepts algébriques derrière le format Tenseur Train et la procédure de Novikov appliquée aux modèles graphiques.
Ce chapitre est divisé en deux parties. La première présente le format Tenseur Train que nous utilisons tout au long de ce manuscrit. Pour cela, nous avons commencé par faire quelques rappels d'algébre linéaire dans la section 4.2. Nous avons ensuite présenté TT-format d'une loi jointe et sa représentation à l'aide de contraction dans le format Tensor Network dans la section 4.3. Enfin, dans la section 4.4, nous avons présenté le format TT-matrice et TT-vecteur qui sera utilisé dans les chapitres 5 et 7.
La deuxième partie de ce chapitre porte sur la procédure de Novikov. Elle est subdivisée en deux sections : une première 4.5 où la mise au format TT d'une loi jointe est développée. Puis, une seconde section 4.6 où est développée l'approche de Novikov [NROV14] pour le calcul des marginales d'un modèle graphique.

Dans le chapitre 5 qui suit, nous utilisons les concepts présentés dans la première partie de ce chapitre pour transposer et expliciter l'approche de Novikov sur les SBM.

Chapitre 5

Approche de type Tenseur Train pour le calcul des marginales binaires dans un modèle à blocs stochastiques

Table des matières

5.1 Introduction

Dans le chapitre précédant, nous avons expliqué l'approche Tenseur Train pour la séparation des variables et son application dans la procédure de Novikov et al. [NROV14] pour le calcul des marginales unaires dans le cadre d'un modèle graphique. Dans ce chapitre :

- nous avons dans un premier temps présenté une déclinaison de la procédure présentée par Novikov et al. [NROV14] pour le calcul des marginales d'un modèle SBM. Nous avons ensuite présenté les difficultés calculatoires derrière cette approche ainsi qu'une réécriture de cette procédure dans le format tenseur train ;
- ensuite, nous avons présenté une organisation des calculs des marginales sur un arbre ainsi que la complexité de cette organisation des calculs.

5.2 Approche TT appliquée sur un modèle SBM

Nous avons présenté les SBM dans le chapitre 2 comme un cas particulier de modèles graphiques où les facteurs ne concernent que des paires d'individus, et pour toutes les paires. Nous pouvons donc parler de liens au lieu d'hyperliens. Cela aura pour effet la simplification de la procédure de Novikov et plus précisément l'étape 1 : En effet, les facteurs sont binaires donc nous pouvons utiliser des matrices au lieu des tenseurs pour les stocker. De plus, la TT-SVD se transforme en une simple SVD.
De plus, dans le cadre des SBM, chaque individu est connecté avec tous les autres. Le niveau d'interaction est une clique. Il s'agit du cas le plus difficile pour l'inférence dans les modèles graphiques avec des facteurs binaires. L'impact de cela et que nous devons traiter m facteurs binaires et n facteurs unaires avec m qui s'exprime :

$$m = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1 = \frac{n(n-1)}{2}$$

Nous allons faire la présentation de la méthode de calcul des marginales dans le cadre d'un modèle SBM Poisson. Cette procédure pourra se généraliser pour les SBM pondérés par des distances.

5.2.1 Rappels sur les SBM

Paramètres des modèles : Un modèle SBM prend en entrée une matrice de distances $n \times n$ si le nombre d'individus est $n \in \mathbb{N}$.
Il prend comme paramètres : un nombre de classes $Q \in \mathbb{N}$, une matrice de distance $D \in \mathbb{M}_{n,n}(\mathbb{N})$, un vecteur de proportions des individus dans les classes $\alpha \in [\![0,1]\!]^Q$ et une structure des distances entre les classes $\Lambda \in \mathbb{M}_{Q,Q}(\mathbb{R})$. Il utilise ces paramètres afin d'exprimer la loi jointe.
L'ensemble des paramètres est désigné par $\theta = (Q, \alpha, \lambda)$.

Loi jointe : Soit $\mathbb{P}_\theta(z_1, \ldots, z_n \mid D)$ la fonction qui exprime la loi jointe conditionnellement aux données du modèle. Alors,

$$\mathbb{P}_\theta(z_1, \ldots, z_n \mid D) = \frac{1}{W} \left[\prod_{i=1}^{n} \prod_{j>i} \psi_{i,j}(z_i, z_j) \right] \left[\prod_{i=1}^{n} \phi_i(z_i) \right] \tag{5.2.1}$$

avec $\psi_{i,j}(z_i, z_j)$ le facteur binaire entre l'individu i et l'individu j et $\phi_i(z_i)$ le facteur unaire de l'individu i.

Dans la formule (5.2.1), il s'agit de la loi jointe conditionnelle, le lien avec la formule (2.2.17) peut être fait par la formule de Bayes.

$$\mathbb{P}(z_1,\ldots,z_n \mid D) = \frac{\mathbb{P}(z,D)}{\mathbb{P}(D)} \tag{5.2.2}$$

La constante de normalisation de $\mathbb{P}(z_1,\ldots,z_n \mid D)$ est égale à $W = \mathbb{P}(D)$.

5.2.2 Calcul des facteurs des modèles

Facteurs Nous commençons par rappeler les formules des facteurs unaires puis binaires du modèle que nous avons présenté dans les formules (2.2.19) et (2.2.18) de la section 2.2.1.4 :

$$\phi_i(z_i) = \Pi_{q=1}^{Q} \alpha_q^{z_{i,q}} \tag{5.2.3}$$

$$\psi_{i,j}(z_i,z_j) = \sum_{q=1}^{Q}\sum_{q'=1}^{Q} \frac{\lambda_{q,q'}^{D(i,j)}}{D(i,j)!} \exp(-\lambda_{q,q'}) z_{i,q} \times z_{j,q'} \tag{5.2.4}$$

avec z_i représente l'état du nœud i et $z_{i,q}$ s'exprime :

$$\begin{cases} z_{i,q} = \mathbb{1}(i \in q) \\ \sum_{q=1}^{Q} z_{i,q} = 1 \end{cases}$$

Nouvelle réécriture des facteurs : Nous décidons de rattacher le facteur unaire $\phi_i(z_i)$ au facteur binaire $\psi_{i,j}(z_i,z_j)$ pour former de nouveaux facteurs binaires. Cela se fait ainsi :

$$\left| \begin{array}{lcll} \psi'_{i,i+1}(z_i,z_{i+1}) & = & \psi_{i,i+1}(z_i,z_{i+1})\phi_i(z_i) & \forall 1 \leq i \leq n-2 \\ \psi'_{i,j}(z_i,z_j) & = & \psi_{i,j}(z_i,z_j) \quad \text{si} \quad j \neq i+1 & \\ \psi'_{n-1,n}(z_{n-1},z_n) & = & \psi_{n-1,n}(z_{n-1},z_n)\phi_{n-1}(z_{n-1})\phi_n(z_n) & \end{array} \right. \tag{5.2.5}$$

En faisant cette transformation, la loi jointe conditionnellement aux données s'exprime :

$$\mathbb{P}_\theta(z_1,\ldots,z_n \mid D) = \frac{1}{W}\prod_{i=1}^{n}\prod_{j>i} \psi'_{ij}(z_i,z_j \mid \theta, D) \tag{5.2.6}$$

Dans ce qui va suivre, pour être en accord avec la section **??**, où les facteurs n'ont qu'un seul indice, nous noterons les tuples $\{i,j\} = \ell$, ainsi, $\ell = 1,\ldots,m$ avec $m = \frac{n(n-1)}{2}$.

Correspondance entre ℓ et i,j : Nous définissons une bijection entre (i,j) et ℓ selon la fonction ζ suivante :

$$\begin{array}{rcl} \zeta : [\![1,n-1]\!] \times [\![i+1,n]\!] & \xrightarrow{\zeta} & [\![1,m]\!] \\ (i,j) & \longrightarrow & (i-1)(n-\frac{i}{2})+j-i \end{array} \tag{5.2.7}$$

Nous prenons un exemple où $n = 6$, si nous choisissons $i = 1, j = 2$. Nous aurons $\ell = 1$ et pour $i = 3, j = 5$ nous aurons $\ell = 11$. Cette correspondance sera utilisée pour le reste du chapitre. Nous exprimons la loi jointe conditionnelle avec la nouvelle notation :

$$\mathbb{P}_\theta(z_1,\ldots,z_n,D) = \frac{1}{W}\prod_{\ell=1}^{m} \psi'_\ell(z^\ell \mid \theta, D) \tag{5.2.8}$$

5.2.3 SVD sur les facteurs binaires et séparation des variables

Soit un tuple d'individus $(i,j) \in [\![1, n-1]\!] \times [\![i+1, n]\!]$. Le facteur binaire associé est $\psi'_{i,j} \in \mathbb{R}^{Q\times Q}$. Avec $\forall z_i, z_j \in [\![1, Q]\!]^2$, les états des individus i et j nous avons $\psi'_{i,j}(z_i, z_j) \in \mathbb{R}$.
Nous faisons une décomposition en valeurs singulières (présentée dans la section 4.2) de $\psi'_{i,j}$:

$$\underbrace{\psi'_{i,j}}_{\mathbb{R}^{Q\times Q}} = \underbrace{U^{i,j}\Sigma^{i,j}}_{G_i^{ij}}\underbrace{(V^{i,j})^T}_{G_j^{ij}} \tag{5.2.9}$$

$$\text{avec, } \begin{cases} G_i^{ij} \in \mathbb{R}^{Q\times r} \\ G_j^{ij} \in \mathbb{R}^{r\times Q} \end{cases}$$

$\Sigma^{i,j}$ est une matrice diagonale qui s'exprime :

$$\Sigma^{i,j} = \begin{pmatrix} \sigma_1 & 0 & \dots & \dots \\ 0 & \sigma_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \sigma_Q \end{pmatrix}$$

Cette décomposition peut se faire de manière exacte avec une SVD complète comme présenté dans la définition 4.2 ou avec une approximation de rang faible comme présenté dans la formule (4.2.9).
Ainsi,

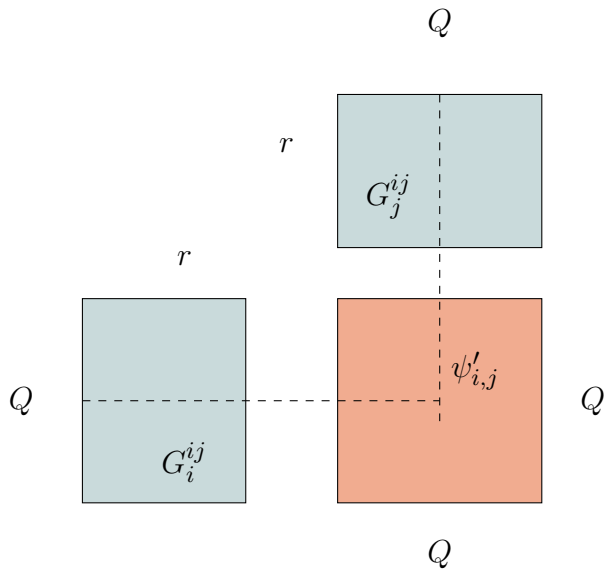


FIGURE 5.1 – Décomposition en valeurs singulières d'un facteur

Aussi :

$$\psi'_{i,j}[z_i, z_j] = \sum_{s=1}^{r} G_i^{i,j}[z_i, s] G_j^{i,j}[s, z_j]$$

Si nous posons :

$$g_i^{i,j}[z_i] = \begin{pmatrix} G_i^{i,j}[z_i, 1] & G_i^{i,j}[z_i, 2] & \dots & G_i^{i,j}[z_i, r] \end{pmatrix} \tag{5.2.10}$$

Et ,

$$g_j^{i,j}[z_j] = \begin{pmatrix} G_j^{i,j}[z_j, 1] \\ G_j^{i,j}[z_j, 2] \\ \vdots \\ G_j^{i,j}[z_j, r] \end{pmatrix} \tag{5.2.11}$$

Ainsi,

$$\underbrace{\psi'_{i,j}(z_i,z_j)}_{\in\mathbb{R}} = \underbrace{g_i^{i,j}[z_i]}_{\in\mathbb{R}^{1\times r}} . \underbrace{g_j^{i,j}[z_j]}_{\in\mathbb{R}^{r\times 1}}$$

Remarque : Nous indiquons dans la figure 5.1 que grâce à la SVD, nous pouvons décomposer les matrices des facteurs $\psi'_{i,j}$ en un produit de deux matrices, G_i^{ij} où chacune ne dépend que de i ou de j. Cela est une séparation des variables pour les facteurs binaires, et sera exploité dans la suite de ce travail pour le calcul de la constante de normalisation et des marginales d'un modèle SBM.

5.2.4 Calcul des matrices intermédiaires $A_i[z_i]$ et B_i :

- **Étape 1 : SVD sur les matrices des nouveaux facteurs binaires** Pour chaque facteur binaire $\psi'_{i,j} \in \mathbb{R}^{Q\times Q}$ associé à l'arête entre deux individsus i et j nous faisons une SVD au rang r :

$$\psi'_{i,j} = U^{i,j}\Sigma^{i,j}(V^{i,j})^T + E^{i,j} \tag{5.2.12}$$

  Avec $U^{i,j} \in R^{Q\times r}$, $\Sigma^{i,j} \in R^{r\times r}$ et $(V^{i,j})^T \in R^{r\times Q}$ où $r \in [\![1,Q]\!]$ le rang de l'approximation de la matrice $\psi'_{i,j}$ avec une erreur $E^{i,j} \in \mathbb{R}^{Q\times Q}$. En particulier, $||E^{i,j}||_F = 0$ si $r = Q$.

  Nous notons :

$$G_i^{i,j} = U^{i,j}\Sigma^{i,j} \tag{5.2.13}$$

  Et,

$$G_j^{i,j} = (V^{i,j})^T \tag{5.2.14}$$

  Avec $G_i^{i,j} \in \mathbb{R}^{Q\times r}$ et $G_j^{i,j} \in \mathbb{R}^{r\times Q}$.
  Donc,

$$\psi'_{i,j} = G_i^{i,j}G_j^{i,j} + E^{i,j} \tag{5.2.15}$$

- **Étape 2 : Ajout des dimensions non essentielles :** Les SBM sont des modèles graphiques où les liens sont binaires. Alors, chaque facteur ne dépend que de deux individus $i,j \in [\![1,n]\!]$. Nous devrons donc rajouter $n-2$ dimensions non essentielles.
  Nous notons $\overline{\psi'_{i,j}}[z_i,z_j]$ l'expression sur les facteurs $\psi'_{i,j}[z_i,z_j]$ avec les dimensions non essentielles.
  Alors la formule (4.6.7) s'exprime :

$$\overline{\psi'_{i,j}}[z_i,z_j] = \underbrace{G_1^{i,j}[z_1]\dots G_{i-1}^{i,j}[z_{i-1}]}_{1}\ \underbrace{G_i^{i,j}[z_i]}_{\_\_\ \in\mathbb{R}^{1,r}}\ \underbrace{G_{i+1}^{i,j}[z_{i+1}]\dots G_{j-1}^{i,j}[z_{j-1}]}_{\mathbb{I}_r}\underbrace{G_j^{i,j}[z_j]}_{|\ \in\mathbb{R}^{r,1}}\underbrace{G_{j+1}^{i,j}[z_{j+1}]\dots G_n^{i,j}[z_n]}_{1} \tag{5.2.16}$$

  où $\_\_$ désigne un vecteur horizontal et $|$ un vecteur vertical. Ainsi, (4.6.8) peut être formulée :

$$\left|\begin{array}{lll} k<i & \Rightarrow & G_k^{i,j}[z_k] = 1 \qquad\qquad \in\mathbb{R}^{1\times 1} \\ k=i & \Rightarrow & G_k^{i,j}[z_k] = G_i^{i,j}[z_i] \quad \in\mathbb{R}^{1\times r} \\ i<k<j & \Rightarrow & G_k^{i,j}[z_k] = \mathbb{I}_{r,r} \qquad\quad \in\mathbb{R}^{r\times r} \\ k=j & \Rightarrow & G_k^{i,j}[z_k] = G_j^{i,j}[z_j] \quad \in\mathbb{R}^{r\times 1} \\ k>i & \Rightarrow & G_k^{i,j}[z_k] = 1 \qquad\qquad \in\mathbb{R}^{1\times 1} \end{array}\right. \tag{5.2.17}$$

- **Étape 4 : Séparation des variables :** Nous calculons les matrices intermédiaires $A_i[z_i]$ et B_i comme nous l'avons fait durant l'étape 4 du procédé de Novikov. Pour cela, nous utiliserons les matrices calculées à la fin des deux premières étapes pour construire la matrice $A_i[z_i]$ de cette manière :

	$i=1$	$i=2$	$\dots$	$i=n$
$\psi'_{1,2}$	$G_1^{1,2}[z_1]$	$G_2^{1,2}[z_2]$	$\dots$	$G_n^{1,2}[z_n]$
$\psi'_{1,3}$	$G_1^{1,3}[z_1]$	$G_2^{1,3}[z_2]$	$\dots$	$G_n^{1,3}[z_n]$
$\vdots$			$\vdots$	
$\psi'_{i,j}$	$G_1^{i,j}[z_1]$	$G_2^{i,j}[z_2]$	$\dots$	$G_n^{i,j}[z_n]$
$\vdots$			$\vdots$	
$\psi'_{n-1,n}$	$G_1^{n-1,n}[z_1]$	$G_2^{n-1,n}[z_2]$	$\dots$	$G_n^{n-1,n}[z_n]$
$\Rightarrow$	$A_1[z_1]$	$A_2[z_2]$	$\dots$	$A_n[z_n]$

Avec,

$$A_i[z_i] = G_i^{1,2}[z_i] \otimes_k G_i^{1,3}[z_i] \otimes_k \dots \otimes_k G_i^{n-1,n}[z_i] \tag{5.2.18}$$

où $\otimes_k$ désigne le produit de Kronecker.
On appelle les $\forall \ell \in [\![1,n]\!], k \in [\![\ell+1,n]\!]$ $G_i[z_i]$ les cores de la matrice $A_i[z_i]$.

Puis, les matrices B_i s'expriment :

$$B_i = \sum_{z_i} A_i[z_i] \tag{5.2.19}$$

5.2.5 Tailles des matrices intermédiaires $A_i[z_i]$ et B_i

Dans cette section, nous présenterons quelques propriétés des matrices $A_i[z_i]$ et B_i et les problèmes d'implémentation que leur manipulation pourrait engendrer.

Proposition 5. *Soit $i \in [\![1,n]\!]$, la matrice $A_i[z_i]$ est calculée à partir de*

- P_1 **:** $i-1$ *cores de tailles* $r \times 1$
- P_2 **:** $n-i$ *cores de tailles* $1 \times r$
- P_3 **:** $(i-1)(n-i)$ *cores de tailles* $r \times r$
- P_4 **:** $\frac{(n-1)(n-2)}{2} - (i-1)(n-i)$ *cores de tailles* 1×1

Démonstration. Soit $n \in \mathbb{N}$, $m = \frac{n(n-1)}{2}$ et $i \in [\![1,n]\!]$. La matrice $A_i[z_i]$ s'exprime :

$$A_i[z_i] = G_i^{1,2}[z_i] \otimes_k G_i^{1,3}[z_i] \otimes_k \dots \otimes_k G_i^{n-1,n}[z_i]$$

Cette matrice est formée par le produit de Kronecker de m cores qui sont soit des scalaires, soit des vecteurs soit des matrices.
Nous nous intéressons maintenant aux tailles des cores.
Le nœud i apparait dans les liens suivants :

- Par la droite : $1 \to i, 2 \to i, \dots, i-1 \to i$

- Par la gauche : $i \to i+1, i \to i+2, \dots, i \to n$

On utilise la formule (5.2.17) pour déterminer la tailles des cores :

$$A_i[z_i] = G_i^{1,2}[z_i] \otimes_k G_i^{1,3}[z_i] \otimes_k \ldots \otimes_k G_i^{n-1,n}[z_i]$$

$$\text{avec,} \quad \left| \begin{array}{lll} i < k & \Rightarrow & G_i^{k,\ell}[z_i] = 1 & \in \mathbb{R}^{1\times 1} \\ i = k & \Rightarrow & G_i^{k,\ell}[z_i] = G_i^{i,l}[z_i] & \in \mathbb{R}^{1\times r} \\ k < i < \ell & \Rightarrow & G_i^{k,\ell}[z_i] = \mathbb{I}_{r,r} & \in \mathbb{R}^{r\times r} \\ i = l & \Rightarrow & G_i^{k,\ell}[z_i] = G_i^{k,i}[z_i] & \in \mathbb{R}^{r\times 1} \\ i > \ell & \Rightarrow & G_i^{k,\ell}[z_i] = 1 & \in \mathbb{R}^{1\times 1} \end{array} \right. \tag{5.2.20}$$

- Quand i apparaît à droite d'une connexion : Cela donne des cores de tailles $r \times 1$: Nous remarquons que nous avons $i-1$ connexion par la droite et donc $i-1$ cores de tailles $r \times 1$.(P_1)
- Quand i apparaît à gauche d'une connexion : Cela donne des cores de tailles $1 \times r$: Comme nous avons $n-i$ connexion à gauche donc $n-i$ cores de tailles $1 \times r$. (P_2)
  Á partir des deux premières propriétés, nous savons que nous avons $m-(n-1) = \frac{(n-1)(n-2)}{2}$ cores scalaires ou identités
  Nous utilisons la formule (5.2.20) pour constater que les cores identités sont :

  $$\underbrace{G_i^{1,i+1}[z_i], G_i^{1,i+2}[z_i], \ldots G_i^{1,n}[z_i]}_{n-i}, \underbrace{G_i^{2,i+1}[z_i], G_i^{2,i+2}[z_i] \ldots G_i^{2,n}[z_i]}_{n-i}, \underbrace{\ldots}_{i-4} \underbrace{G_i^{i-1,i+1}[z_i], \ldots G_i^{i-1,n}[z_i]}_{n-i}$$

  Donc, il y aurait tout simplement $(i-1)(n-i)$ cores identités. (P_3)
- Á partir de P_1, P_2 et P_2, nous aurons $\frac{(n-1)(n-2)}{2} - i(n-i)$ cores de tailles $1 \times 1.P_4$

□

Á partir de cette propriété, nous pouvons déduire les tailles des matrices.

Corollaire 6. *Les matrices $A_i[Z_i]$ et B_i sont de tailles $r^{(i-1)(n-i+1)} \times r^{(i)(n-i)}$.*

Démonstration. On utilise la proposition 9 pour déterminer la taille des matrices $A_i[z_i]$. En effet, comme ces matrices sont formées par les produits de $i-1$ cores de tailles $1 \times r$, $n-i$ cores de tailles $r \times 1$ et $(i-1)(n-i)$ cores de tailles $r \times r$.

Alors la taille de $A_i[z_i]$ est :

$$r^{(i-1)+(i-1)(n-i)} \times r^{(n-i)+(i-1)(n-i)}$$

Ainsi, les $A_i[z_i]$ sont de tailles $r^{(i-1)(n-i+1)} \times r^{(i)(n-i)}$.

Donc, comme $B_i = \sum_{z_i} A_i[z_i]$,
Alors B_i sont de tailles $r^{(i-1)(n-i-1)} \times r^{(i)(n-i)}$. □

Remarque 7. *Quelques remarques par rapport aux tailles des matrices :*

- *$A_1[z_1]$ et B_1 sont des vecteurs de tailles $1 \times r^{(n-1)}$.*
- *$A_n[z_n]$ et B_n sont des vecteurs de tailles $r^{(n-1)} \times 1$.*
- *Les matrices $A_i[z_i]$ et B_i sont de tailles exponentielles, il est donc difficile de les stocker et les manipuler.*

5.2.6 Réécriture de la procédure dans le format tenseur train

Une manière possible de traiter ce type de matrices est de les traiter comme des matrices creuses [GVL96], cependant les tailles de ces matrices restent imenses.
Oseledets [Ose11] a développé les méthodes pour manipuler ces matrices grâce à la tensorisation (quantisation) qui permet de les écrire sous la forme de TT-matrices que nous avons présentées dans la définition 4.4.2 dans la section 4.4.2. Cette méthode nous permet de stocker ces matrices de tailles $\mathcal{O}(r^n \times r^n)$ en $\mathcal{O}(n^2 Q r^2)$.
Dans ce qui va suivre, nous expliquerons comment faire les calculs dans ce format.

NB : Dans cette section, nous allons suivre la même numérotation des étapes que les sections précédentes. De plus, nous noterons les tenseurs en gras et en majuscule.
Á partir de l'étape 4, nous manipulerons les matrices $A_i[z_i]$ sous format TT (TT-matrices) que nous avons présenté dans la définition 4.4.2.

Objectif : Passage du format matriciel de $A_i[z_i]$ à un format TT matrice. C'est-à-dire l'écriture de ces matrices sous la forme que nous avons définie durant la section 4.4.2

$$\forall \mathbf{i} \in [\![1, r^{(i-1)(n-i+1)}]\!] = (i_1, \dots, i_m), \quad \mathbf{j} \in [\![1, r^{i(n-i)}]\!], \quad h \in [\![1, m]\!] = (j_1, \dots, j_m),$$
$$A_i[z_i](\mathbf{i}, \mathbf{j}) = \mathbf{A}_i[z_i](i_1, \dots, i_m; j_1, \dots, j_m) = \mathbf{G}_i^{1,2}[z_i](i_1, j_1)\mathbf{G}_i^{1,3}[z_i](i_2, j_2) \dots \mathbf{G}_i^{n-1,n}[z_i](i_m, j_m)$$

Nous utilisons la fonction définie dans la formule (5.2.7) et nous exprimons e comme la solution de :

$$e = \zeta(k, \ell)$$

Alors,

$$\left| \begin{array}{llllll} i < k & \Rightarrow & \mathbf{G}_i^{k,\ell}[z_i](i_e, j_e) = 1 & \in \mathbb{R} & i_e = 1 & j_e = 1 \\ i = k & \Rightarrow & \mathbf{G}_i^{k,\ell}[z_i](i_e, j_e) = G_i^{i,l}[z_i] & \in \mathbb{R}^{1\times r} & i_e = 1 & j_e = 1, \dots, r \\ k < i < \ell & \Rightarrow & \mathbf{G}_i^{k,\ell}[z_i](i_e, j_e) = \mathbb{I}_{r,r} & \in \mathbb{R}^{r\times r} & i_e = 1, \dots, r & j_e = 1, \dots, r \\ i = l & \Rightarrow & \mathbf{G}_i^{k,\ell}[z_i](i_e, j_e) = G_i^{k,i}[z_i] & \in \mathbb{R}^{r\times 1} & i_e = 1, \dots, r & j_e = 1 \\ i > \ell & \Rightarrow & \mathbf{G}_i^{k,\ell}[z_i](i_e, j_e) = 1 & \in \mathbb{R} & i_e = 1 & j_e = 1 \end{array} \right. \tag{5.2.21}$$

Si nous écrivons chaque coefficient de cette TT-matrice comme une contraction de tenseurs $\mathbf{G}_i^{k,l}[z_i]$ d'ordre 4 comme présenté dans 4.4.2 (les deux formes, matricielles et tensorielles, sont équivalents), nous obtenons :

$$\mathbf{A}_i[z_i] = \mathbf{G}_i^{1,2}[z_i] \bullet_{\alpha_1} \mathbf{G}_i^{1,3}[z_i] \bullet_{\alpha_2} \dots \bullet_{\alpha_{d-2}} \mathbf{G}_i[z_i]^{n-2,n} \bullet_{\alpha_{d-1}} \mathbf{G}_i[z_i]^{n-1,n} \tag{5.2.22}$$

En le développant selon les quatre indices, nous obtenons pour chaque coefficient :

$$\begin{aligned} \mathbf{A}[z_i][i_1, \dots, i_d; j_1, \dots, j_d] = \sum_{\alpha_1=1}^{r_1} \dots \sum_{\alpha_{d-1}=1}^{r_{d-1}} & \mathbf{G}_i^{1,2}[z_i](1, i_1, j_1, \alpha_1)\mathbf{G}_i^{1,3}[z_i](\alpha_1, i_2, j_2, \alpha_2) \\ & \dots \mathbf{G}_i[z_i]^{n-2,n}(\alpha_{m-2}, i_{m-1}, j_{m-1}, \alpha_{m-1})\mathbf{G}_i[z_i]^{n-1,n}(\alpha_{m-1}, i_m, j_m, 1) \end{aligned} \tag{5.2.23}$$

Avec,
$$\left| \begin{array}{llll} i < k & \Rightarrow & \mathbf{G}_i^{k,\ell}[z_i] = 1 & \in \mathbb{R}^{1\times 1\times 1\times 1} \\ i = k & \Rightarrow & \mathbf{G}_i^{k,\ell}[z_i] = G_i^{i,l}[z_i] & \in \mathbb{R}^{1\times 1\times r\times 1} \\ k < i < \ell & \Rightarrow & \mathbf{G}_i^{k,\ell}[z_i] = \mathbb{I}_{r,r} & \in \mathbb{R}^{1\times r\times r\times 1} \\ i = l & \Rightarrow & \mathbf{G}_i^{k,\ell}[z_i] = G_i^{k,i}[z_i] & \in \mathbb{R}^{1\times 1\times r\times 1} \\ i > \ell & \Rightarrow & \mathbf{G}_i^{k,\ell}[z_i] = 1 & \in \mathbb{R}^{1\times 1\times 1\times 1} \end{array} \right. \tag{5.2.24}$$

Ces TT-matrices $A_i[z_i]$ sont de rang 1.

- **Étape 1 : Tensorisation des quantités $G_i^{i,j}$ et $G_j^{i,j}$:** Cette étape s'appuie sur l'étape 1 de la section 5.2.4 . En effet, après le calcul de la SVD, nous obtenons $G_i^{i,j} \in \mathbb{R}^{Q\times r}$ et $G_j^{i,j} \in \mathbb{R}^{r\times Q}$.
  Durant cette étape, nous faisons la transformation de ces matrices en tenseur afin de créer les premiers cores des TT-matrices $A_i[z_i]$ comme nous avons expliqué dans la section 4.4.2.

$$\mathbf{G}_i^{i,j} \in \mathbb{R}^{1\times Q\times r\times 1} \qquad \text{et} \qquad \mathbf{G}_j^{i,j} \in \mathbb{R}^{1\times r\times Q\times 1} \tag{5.2.25}$$

  Avec,

$$\mathbf{G}_i^{i,j}[z_i] \in \mathbb{R}^{1\times 1\times r\times 1} \qquad \text{et} \qquad \mathbf{G}_j^{i,j}[z_j] \in \mathbb{R}^{1\times r\times 1\times 1} \tag{5.2.26}$$

  Dans ce format,

$$\underbrace{\psi'_{i,j}[z_i,z_j]}_{\in\mathbb{R}} = \underbrace{\mathbf{G}_i^{i,j}(1,z_i,:,1)}_{\in\mathbb{R}^{1,r}}\underbrace{\mathbf{G}_j^{i,j}(1,:,z_j,1)}_{\in\mathbb{R}^{r,1}} \tag{5.2.27}$$

- **Étape 2 : Tensorisation des dimensions non essentielles :** L'implémentation de la manière avec laquelle se fait le rajout des dimensions non essentielles s'appuie sur l'étape 2 de la section 5.2.4 où nous faisons l'ajout des dimensions non essentielles. Puis, pour tous les tuples $\{i,j,k\} \in [\![1,n]\!]^3$, nous transformons les matrices G_k^{ij} en tenseurs $\mathbf{G}_k^{ij}$.
  Ainsi, nous aurons :

$$\overline{\psi'}_{i,j}(z_i,z_j) = \prod_{k=1}^{n} \mathbf{G}_k^{i,j}(z_k) \tag{5.2.28}$$

  Á partir de la formule (5.2.24), nous obtenons les dimensions des cores :

$$\begin{aligned}\underbrace{\overline{\psi'}_{i,j}(z_i,z_j)}_{\in\mathbb{R}} &= \underbrace{\mathbf{G}_1^{ij}(1,1,z_1,1)}_{\in\mathbb{R}}\dots\underbrace{\mathbf{G}_{i-1}^{ij}(1,1,z_{i-1},1)}_{\in\mathbb{R}}\underbrace{\mathbf{G}_i^{i,j}(1,z_i,:,1)}_{\in\mathbb{R}^{1,r}}\\ &\underbrace{\mathbf{I}_{1,r,Q,r}(1,:,z_{i+1},:)}_{\mathbb{R}^{r\times r}}\dots\underbrace{\mathbf{I}_{1,r,r,Q}(1,:,z_{j-1},:)}_{\mathbb{R}^{r\times r}}\underbrace{\mathbf{G}_j^{i,j}(1,:,z_j,1)}_{\in\mathbb{R}^{r,1}}\underbrace{\mathbf{G}_{j+1}^{ij}(1,z_{j+1},1,1)}_{\in\mathbb{R}}\dots\underbrace{\mathbf{G}_n^{ij}(1,z_n,1,1)}_{\in\mathbb{R}}\end{aligned} \tag{5.2.29}$$

  Avec,

$$\mathbf{I}_{1,r,Q,r}(1,:,Z_i=k,:) = \mathbb{I}_{r,r} \tag{5.2.30}$$

  Les transformations en tenseurs des matrices G_k^{ij} présentées dans (5.2.17) s'exprime :

$$\left|\begin{array}{lll} & & \\ k<i & \Rightarrow & \mathbf{G}_k^{ij} = \overbrace{\left(1,\dots,1\right)}^{\text{Q lignes}} \quad \in\mathbb{R}^{1\times 1\times Q\times 1}\\ k=i & \Rightarrow & \mathbf{G}_i^{ij} = \mathbf{G}_i^{ij} \quad \in\mathbb{R}^{1\times Q\times r\times 1}\\ i<k<j & \Rightarrow & \mathbf{G}_k^{ij} = \mathbf{I}_{1,r,Q,r} \quad \in\mathbb{R}^{1\times r\times Q\times r}\\ k=j & \Rightarrow & \mathbf{G}_j^{ij} = \mathbf{G}_j^{ij} \quad \in\mathbb{R}^{1\times r\times Q\times 1}\\ & & \\ k>j & \Rightarrow & \mathbf{G}_k^{ij} = \overbrace{\left(1,\dots,1\right)}^{\text{Q lignes}} \quad \in\mathbb{R}^{1\times Q\times 1\times 1}\end{array}\right. \tag{5.2.31}$$

- **Étape 4 : Calcul des TT-matrices intermédiaires $\mathbf{A}_i[z_i]$ et $\mathbf{B}_i$:** Durant l'étape 4 de la section 5.2.4, nous avons exprimé les $A_i[z_i]$ sous format matriciel. Dans cette étape, nous

allons les exprimer sous format tensoriel. Pour chaque individu $i \in [\![1, n]\!]$, nous construisons les TT-matrices $\mathbf{A}_i[z_i]$ à partir des m cores $[\mathbf{G}^{1,2}[z_i]\mathbf{G}^{1,3}[z_i]\dots\mathbf{G}^{n-1,n}[z_i]]$
Ainsi, les TT-matrices s'expriment :

$$\mathbf{A}_i[z_i](i_1,\dots,i_m;j_1,\dots,j_m) = \mathbf{G}^{1,2}[z_i](i_1,j_1)\mathbf{G}^{1,3}[z_i](i_2,j_2)\dots\mathbf{G}^{n-1,n}[z_i](i_m,j_m) \quad (5.2.32)$$

Ce format nous permettrait de stocker les matrices $A_i[z_i]$ dont les tailles $r^{(i-1)(n-i-1)} \times r^{(i)(n-i)}$ présentées lors du corollaire en mQr^2.
Enfin, les $\mathbf{B}_i$ s'expriment :

$$\mathbf{B}_i = \sum_{z_i} \mathbf{A}_i[z_i] \quad (5.2.33)$$

Remarques :

- Les TT-matrices $A_i[z_i]$ et B_i sont des tenseurs d'ordres m.
- $A_1[z_1]$ et $A_n[z_n]$ sont des TT-vecteurs de rang 1.
- B_1 et B_n sont des TT-vecteurs de rang Q .
- Les TT-matrices $(A_i[z_i])_{i=2,\dots,n-1}$ sont des TT-matrice de rang 1.
- Les TT-matrices $(B_i)_{i=2,\dots,n-1}$ sont les sommes des $A_i[z_i]$, elles sont des TT-matrices de rang Q comme expliqué dans la propriété 3 de la section 4.1.2 du chapitre 4.

- **Étape 5 : Calcul de la constante de normalisation et des marginales :** Les calculs des trois quantités suivantes se font par produits de TT-matrices d'ordre m. Comme nous l'avons présenté dans la propriété 2 de la section 4.1.2 du chapitre 4, les résultats des marginales et de la constante de normalisation sont TT-matrices d'ordres m.
  - Constante de normalisation :
  $$\mathbf{W} = \mathbf{B}_1\dots\mathbf{B}_n$$
  - Marginales unaires : $\forall i \in [\![1, n]\!]$,
  $$\mathbf{p}_i(z_i) = \mathbf{B}_1\dots\mathbf{B}_{i-1}\mathbf{A}_i[z_i]\mathbf{B}_{j+1}\dots\mathbf{B}_n$$
  - Marginales binaires : $\forall i, j \in [\![1, n]\!]^2$,
  $$\mathbf{p}_{i,j}(z_i, z_j) = \mathbf{B}_1\dots\mathbf{B}_{i-1}\mathbf{A}_i[z_i]\mathbf{B}_{i+1}\dots\mathbf{B}_{j-1}\mathbf{A}_j[z_j]B_{j+1}\dots\mathbf{B}_n$$

- **Étape 6 : Passage du format TT aux formats adéquats :** nous avons précisé dans la remarque 10 que $A_1[z_1]$ et $A_n[z_n]$ B_1 et B_n sont des TT-vecteurs. Alors, tous les calculs présentés lors de l'étape 5 seront des produits matrices vecteurs.
  Nous notons :
  $$\mathbf{W}^i = \mathbf{B}_1\dots\mathbf{B}_i$$
  Comme $\mathbf{B}_1$ est un TT-vecteur, alors $\mathbf{W}^i$ sera un TT-vecteur. Si nous faisons une généralisation, $\mathbf{W}^{n-1}$ sera un TT-vecteur.
  On a :
  $$\mathbf{W} = \mathbf{W}^{n-1}\mathbf{B}_n$$
  Comme $\mathbf{B}_n$ est un TT-vecteur, alors $\mathbf{W}$ est un TT-vecteur de rang 1 comme présenté dans les propositions sur les TT-vecteurs.
  Nous pouvons donc l'exprimer comme la formule (4.4.5) qu'on a expliqué dans la section 4.1.2 du chapitre 4. Ainsi,
  $$\mathbf{W}[i_1,\dots,i_m] = \mathbf{G}_1^{\mathbf{W}}(i_1).\mathbf{G}_2^{\mathbf{W}}(i_2).\dots.\mathbf{G}_m^{\mathbf{W}}(i_m)$$

De plus, comme $\mathbf{W}$ est de TT-rang 1, alors : $\forall k \in [\![1,m]\!], i_k = 1$. Et donc $\mathbf{W}$ est un tenseur d'ordre m avec un seul élément. En effet, nous pouvons réaliser que :

$$\mathbf{W}[i_1,\ldots,i_m] = \mathbf{W}[1,\ldots,1] = \underbrace{\mathbf{G}_1^{\mathbf{W}}(1)}_{\in \mathbb{R}^{1\times 1\times r_1\times 1}} . \underbrace{\mathbf{G}_2^{\mathbf{W}}(1).\ldots.\mathbf{G}_{m-1}^{\mathbf{W}}(1).}_{\in \mathbb{R}^{1\times r_{k-1}\times r_k\times 1}} \underbrace{\mathbf{G}_m^{\mathbf{W}}(1)}_{\in \mathbb{R}^{1\times r_{m-1}\times 1\times 1}}$$

Donc, nous pouvons exprimer la constante de normalisation comme un réel.
Cette procédure peut se généraliser pour les marginales unaires et binaires.

5.2.7 Implémentation de l'approche :

Nécessité du rounding pour les calculs : Les calculs des marginales ou la constante de normalisation nécessitent un produit de TT-matrices TT-vecteurs. Or, nous avons expliqué durant 2 que le rang de la TT-matrices résultante augmente.
Comme nous faisons n produit de ce type, les rangs augmenteront de manière exponentielle. Il est donc nécessaire de faire une recompression du résultat à chaque itération via une procédure de compression par le rounding présentée dans la sous-section rounding.
Les calculs se feront de cette manière :

$$\mathbf{W} = \text{rounding}(\text{rounding}(\ldots\text{rounding}(\mathbf{B}_1 \times \mathbf{B}_2, \epsilon) \times \ldots \times \mathbf{B}_{n-1}, \epsilon) \times \mathbf{B}_n, \epsilon)$$

Dans ce cas, le calcul de la constante de normalisation est calculée de cette manière :

- $\mathbf{W}_1 = \mathbf{B}_1$
- Pour $i = 2,\ldots,n$:
  - $\mathbf{W}_i = \mathbf{W}_{i-1} \times \mathbf{B}_i$
  - $\mathbf{W}_i = \text{rounding}(\mathbf{W}_i, \epsilon)$
- $\mathbf{W} = \mathbf{W}_n$

Implémentation des calculs de droite à gauche : Les librairies permettant le calcul via des Tenseurs Trains sont plus adaptés pour des calculs TT-matrices-TT-vecteurs que pour des calculs TT-vecteurs-TT-matrices.
Nous avons donc adapté notre implémentation des calculs en partant de la droite vers la gauche. C'est-à-dire :

$$\mathbf{W} = \text{rounding}(\mathbf{B}_1 \times \text{rounding}(\mathbf{B}_2 \times \ldots \times \text{rounding}(\mathbf{B}_{n-1} \times \mathbf{B}_n, \epsilon), \epsilon), \epsilon)$$

Dans ce cas, le calcul de la constante de normalisation est calculée de cette manière :

- $\mathbf{W}_n = \mathbf{B}_n$
- Pour $i = n-1,\ldots,1$:
  - $\mathbf{W}_i = \mathbf{B}_i \times \mathbf{W}_{i+1}$
  - $\mathbf{W}_i = \text{rounding}(\mathbf{W}_i, \epsilon)$
- $\mathbf{W} = \mathbf{W}_1$

5.3 Méthode de calcul des marginales via une mutualisation des calculs

Dans la librairie TT-MRF, Novikov et al. [NROV14] utilisent le message passing [YFW03] qui consiste à calculer l'ensemble des produits : $\forall i \in [\![1,n]\!], \quad \mathbf{B}_{i+1} \times \ldots \times \mathbf{B}_n$ et $\mathbf{B}_1 \times \ldots \times \mathbf{B}_{i-1}$ puis les utilisent pour le calcul des marginales unaires.

Dans ce qui va suivre, nous présentons une approche similaire pour le calcul des marginales binaires. Nous rappelons auparavant l'approche pour le calcul des marginales unaires.

5.3.1 Calcul des marginales unaires

Soit un modèle SBM de paramètres n, Λ, α et Q. Nous appliquons la procédure de l'approche TT et nous obtenons les TT-matrices $\forall i \in [\![1,n]\!], \mathbf{A}_i[z_i]$ et $\mathbf{B}_i$.
Nous prenons par exemple $i \in [\![1,n-2]\!]$. Le calcul des marginales unaires de i, $i+1$ et $i+2$ s'exprime ainsi :

$$\begin{aligned}
\mathbf{p}_i(z_i) &= \mathbf{B}_1 \times \ldots \times \mathbf{B}_{i-1} \times \mathbf{A}_i[z_i] \times \mathbf{B}_{i+1} \times \ldots \times \mathbf{B}_n \\
\mathbf{p}_{i+1}(z_{i+1}) &= \mathbf{B}_1 \times \ldots \times \mathbf{B}_i \times \mathbf{A}_{i+1}[z_{i+1}] \times \mathbf{B}_{i+2} \times \ldots \times \mathbf{B}_n \\
\mathbf{p}_{i+2}(z_{i+2}) &= \mathbf{B}_1 \times \ldots \times \mathbf{B}_{i+1} \times \mathbf{A}_{i+2}[z_{i+2}] \times \mathbf{B}_{i+3} \times \ldots \times \mathbf{B}_n
\end{aligned} \tag{5.3.1}$$

Les calculs de ces trois marginales unaires pour les individus i, $i+1$ et $i+2$ font appels aux deux mêmes ensembles de produits de matrices : $\mathbf{L}_i = \mathbf{B}_1 \times \ldots \times \mathbf{B}_{i-1}$ et $\mathbf{R}_{i+2} = \mathbf{B}_{j+3} \times \ldots \times \mathbf{B}_n$.
Nous pouvons réécrire les calculs (5.3.1) comme :

$$\begin{aligned}
\mathbf{p}_i(z_i) &= \mathbf{L}_i \mathbf{A}_i[z_i] \mathbf{B}_{i+1} \mathbf{B}_{i+2} \mathbf{R}_{i+2} \\
\mathbf{p}_{i+1}(z_{i+1}) &= \mathbf{L}_i \mathbf{B}_i \mathbf{A}_{i+1}[z_{i+1}] \mathbf{B}_{i+2} \mathbf{R}_{i+2} \\
\mathbf{p}_{i+2}(z_{i+2}) &= \mathbf{L}_i \mathbf{B}_i \mathbf{B}_{i+1} \mathbf{A}_{i+2}[z_{i+2}] \mathbf{R}_{i+2}
\end{aligned} \tag{5.3.2}$$

Il est donc inutile de calculer trois fois les produits $\mathbf{L}_i$ et $\mathbf{R}_{i+2}$. Plus généralement, chaque marginale unaire se décompose en un produit de matrices avec une partie droite et une partie gauche :

$$\forall i \in [\![1,n]\!], \quad \mathbf{p}_i(z_i) = \underbrace{\mathbf{B}_1 \times \ldots \times \mathbf{B}_{i-1}}_{\text{left}} \mathbf{A}_i[z_i] \underbrace{\mathbf{B}_{i+1} \times \ldots \times \mathbf{B}_n}_{\text{right}} \tag{5.3.3}$$

On peut donc exprimer les marginales unaires de cette manière :

$$\mathbf{p}_i(z_i) = \mathbf{L}_i \times \mathbf{A}_i[z_i] \times \mathbf{R}_i \tag{5.3.4}$$

Les parties gauches $\mathbf{L}_i$ et droites $\mathbf{R}_i$ peuvent chacune se calculer par récurrence pour tout $i \in [\![1,n]\!]$ selon les schémas :

$$\begin{cases} \mathbf{L}_1 &= 1 \in \mathbb{R} \\ \mathbf{L}_{i+1} &= \mathbf{L}_i \mathbf{B}_i \qquad \forall i \in [\![2,n]\!] \end{cases} \tag{5.3.5}$$

et

$$\begin{cases} \mathbf{R}_n &= 1 \\ \mathbf{R}_i &= \mathbf{B}_{i+1} \mathbf{R}_{i+1} \qquad \forall i \in [\![1,n-1]\!] \end{cases} \tag{5.3.6}$$

Nous remarquons alors que le calcul de toutes les matrices $\mathbf{L}_i$ et $\mathbf{R}_i$ pour $i \in [\![1,n]\!]$ permet le calcul de toutes les marginales unaires selon la formule 5.3.4 et nécessite $2(n-1)$ produits matriciels seulement. Il en faut donc $2(n-1)+2nQ \sim 2nQ$ pour le calcul de toutes les marginales unaires.

Cette organisation des calculs est connue sous le nom de "message passing", et est efficace dans un cadre plus large de calcul simultané de toutes les marginales pour des lois jointes de modèles graphiques dont le graphe est un arbre [YFW03].

Nous étendons cette approche au calcul de toutes les marginales binaires d'un modèle SBM.

5.3.2 Calcul des marginales binaires

Nous prenons par exemple le couple (i,j). Les calculs des marginales binaires de manière directe s'expriment :

$$\mathbf{p}_{i,j}(z_i,z_j) = \mathbf{B}_1 \times \ldots \times \mathbf{B}_{i-1} \times \mathbf{A}_i[z_i] \times \mathbf{B}_{i+1} \times \ldots \times \mathbf{B}_{j-1}\mathbf{A}_j[z_j]\mathbf{B}_{j+1} \times \ldots \times \mathbf{B}_n \tag{5.3.7}$$

Nous constatons que chaque marginale binaire se décompose en une partie droite, une partie centrale et une partie gauche : $\forall i,j \in [\![1,n]\!]^2$ $\forall z_i,z_j \in [\![1,Q]\!]^2$

$$\mathbf{p}_{i,j}(z_i,z_j) = \underbrace{\mathbf{B}_1 \times \ldots \times \mathbf{B}_{i-1}}_{\text{left}} \times \mathbf{A}_i[z_i] \times \underbrace{\mathbf{B}_{i+1} \times \ldots \times \mathbf{B}_{j-1}}_{\text{center}} \times \mathbf{A}_j[z_j] \times \underbrace{\mathbf{B}_{j+1} \times \ldots \times \mathbf{B}_n}_{\text{right}} \tag{5.3.8}$$

Il existe une partie droite $\mathbf{R}_j$ qui ne dépend que de j, une partie gauche $\mathbf{L}_i$ qui ne dépend que de i et une partie centrale $\mathbf{C}_{i,j}$ qui dépend du couple (i,j).

$$\mathbf{p}_{i,j}(z_i,z_j) = \mathbf{L}_i \times \mathbf{A}_i[z_i] \times \mathbf{C}_{i,j} \times \mathbf{A}_j[z_j] \times \mathbf{R}_j \tag{5.3.9}$$

Les parties gauches $\mathbf{L}_i$ et droites $\mathbf{R}_i$ peuvent chacune se calculer par récurrence pour tout $i \in [\![1,n]\!]$ selon les schémas (5.3.5) et (5.3.6). La partie centrale s'exprime :

$$\forall i \in [\![1,n-2]\!], j \in [\![i+2,n]\!] \qquad \mathbf{C}_{i,j} = \mathbf{B}_{i+1} \times \ldots \times \mathbf{B}_{j-1} \tag{5.3.10}$$

5.3.3 Organisation des calculs sur un arbre

Le calcul indépendant de tous les C_{ij} selon la formule (5.3.10) est très coûteux en termes de produits matriciels. Cependant, si nous connaissons par exemple $\mathbf{C}_{1,4}$, il est simple de calculer $\mathbf{C}_{1,5}$ avec un seul produit matriciel car $\mathbf{C}_{1,5} = \mathbf{C}_{1,4} \times \mathbf{B}_5$. Nous présentons dans ce qui suit une systématisation de cette propriété pour optimiser les calculs de toutes les marginales binaires. Pour cela, nous définissons les quantités $\mathbf{G}$:

$$\forall i \in [\![1,n-1]\!], j \in [\![i+1,n]\!], \qquad \mathbf{G}_{i,j} = \mathbf{B}_i \times \mathbf{B}_{i+1} \times \ldots \times \mathbf{B}_{j-1} \times \mathbf{B}_j \tag{5.3.11}$$

Ces quantités permettront de calculer simultanément les parties $\mathbf{R}_i$, $\mathbf{L}_j$ et $\mathbf{C}_{i,j}$ en organisant la récurrence des calculs sur un arbre. Un tel arbre est illustré dans la figure 5.2 dans le cas $n=5$.

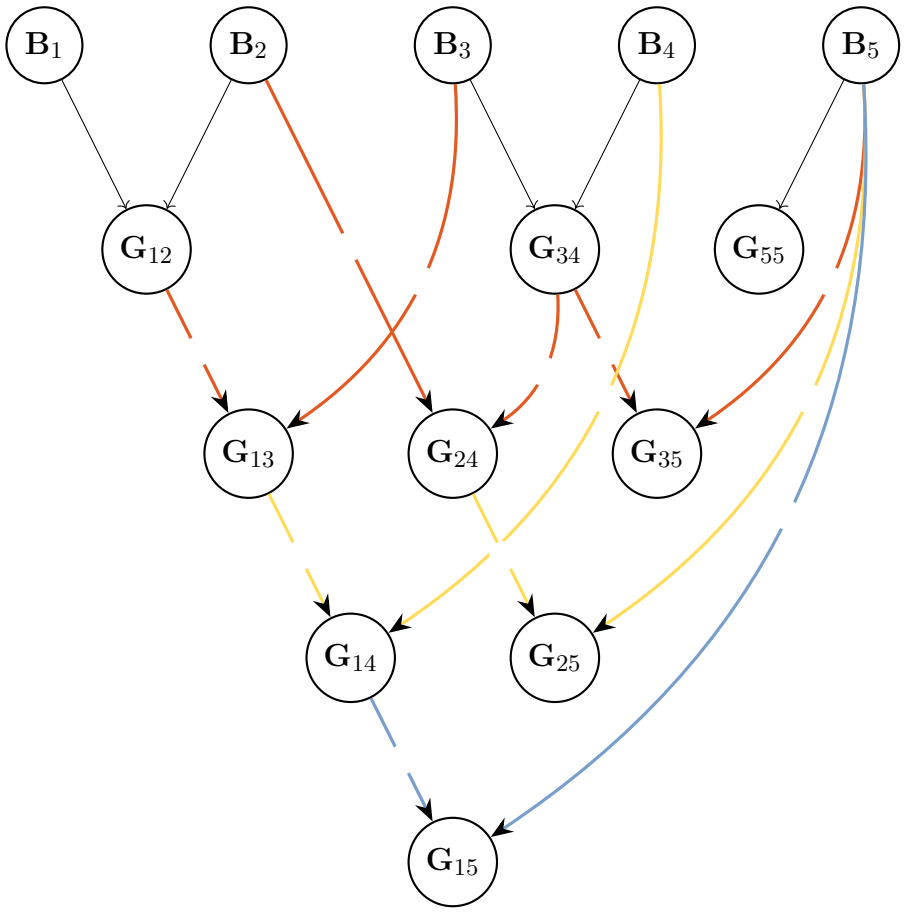


FIGURE 5.2 – Décomposition des calculs des parties centrales de l'expression des marginales binaires

Explication de l'arbre : Chaque nœud interne de cet arbre représente une matrice $\mathbf{G}_{i,j}$. les feuilles sont les matrices $\mathbf{B}_i$. Chaque nœud $\mathbf{G}_{i,j}$ a deux parents : $\mathbf{G}_{i,j-1}$ et $\mathbf{B_j}$. Il est alors le produit de ses parents. Le calcul sur l'ensemble de l'arbre se fait par récurrence en partant des feuilles. Nous représentons la succession des calculs sur la figure 5.2 par des couleurs : les calculs se font dans l'ordre noir, orange, jaune et bleue. In fine, la racine de l'arbre représente la constante de normalisation $\mathbf{W}$.

La récurrence selon le parcours de l'arbre des feuilles à la racine s'écrit :

$$\begin{cases} \mathbf{G_{i,i}} = \mathbf{B}_i & \forall i \in [\![1, n-1]\!] \\ \mathbf{G_{i,j+1}} = \mathbf{G_{i,j}} \times \mathbf{B}_{j+1} & \forall j \in [\![i+1, n]\!] \end{cases} \tag{5.3.12}$$

qui permet le calcul de toutes les parties centrales de l'expression des marginales binaires. En effet, à partir de (5.3.10) et (5.3.11), nous pouvons exprimer les parties centrales $\mathbf{C}_{i,j}$ à partir de $\mathbf{G}_{i,j}$ selon

$$\forall i \in [\![1, n-1]\!], j \in [\![i+1, n]\!] \qquad \mathbf{C}_{i,j} = \mathbf{G}_{i+1,j-1} \tag{5.3.13}$$

De plus, nous obtenons les parties gauches L_i et droites R_i des formules (5.3.9) et (5.3.4) selon :

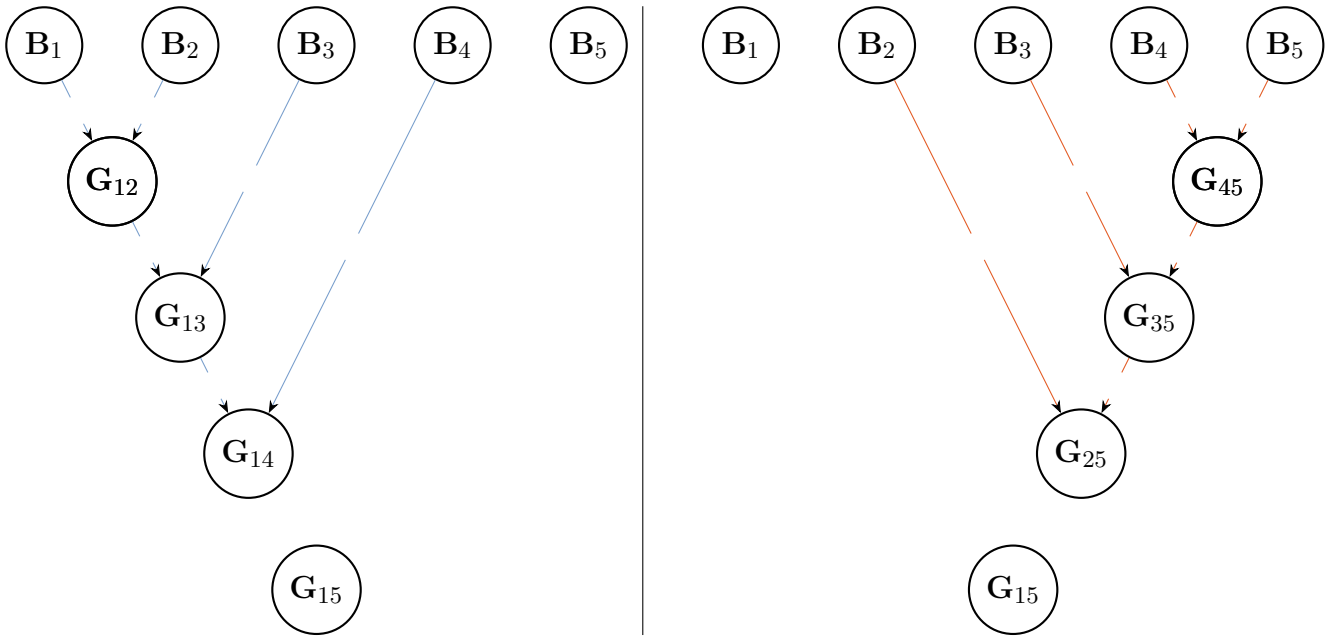


FIGURE 5.3 – Décomposition des calculs de la partie Gauche ($\mathbf{L_i}$) en bleu et la partie droite ($\mathbf{R_j}$) en orange de l'expression des marginales binaires

soit,

$$\begin{cases} \mathbf{L}_1 &= 1 \in \mathbb{R} \\ \mathbf{L}_2 &= \mathbf{B}_1 \\ \mathbf{L}_{i+1} &= \mathbf{G}_{1,i} \qquad \forall i \in [\![3, n-1]\!] \end{cases} \tag{5.3.14}$$

$$\begin{cases} \mathbf{R}_n &= 1 \in \mathbb{R} \\ \mathbf{R}_{n-1} &= \mathbf{B}_n \\ \mathbf{R}_{n-j} &= \mathbf{G}_{j,n} \qquad \forall i \in [\![2, n-1]\!] \end{cases} \tag{5.3.15}$$

Complexité des calculs sur l'arbre Nous généralisons l'arbre de la figure 5.2 pour $n \in \mathbb{N}$ individus pour le calcul de tous les nœuds de l'arbre. Il faut calculer $n-1$ produits matriciels pour la première itération, $n-2$ produits matriciels pour la seconde itération, ... et un produit matriciel pour la dernière itération. Une itération $i \in [\![1, n-1]\!]$ donnée nécessite le calcul de $n-i$ produits matriciels. Donc, l'ensemble des calculs nécessite π_n produits matriciels, avec :

$$\pi_n = \sum_{i=1}^{n-1} (n-i) = \frac{n(n-1)}{2}$$

Les marginales binaires se calculent selon :

$$p_{i,j}(z_i, z_j) = \underbrace{\mathbf{L}_i \times \mathbf{A}_i[Z_i] \times \mathbf{C}_{i,j} \times \mathbf{A}_j[Z_j] \times \mathbf{R}_j}_{\text{4 produits matriciels}}$$

Connaissant $\mathbf{L}_i$, $\mathbf{C}_{i,j}$ et $\mathbf{R}_j$, le calcul de l'ensemble des marginales binaires nécessite $2 \times n(n-1) \times Q^2$ produits matriciels.

Ainsi, le calcul de toutes les marginales nécessite S_n produits matriciels, avec

$$S_n = 2 \times n(n-1) \times Q^2 + \frac{n(n-1)}{2} = n(n-1)\left(2Q^2 + \frac{1}{2}\right) \approx 2n(n-1)Q^2 \sim \mathcal{O}(n^2 Q^2) \tag{5.3.16}$$

5.4 Conclusion

Dans ce chapitre, nous avons en premier lieu présenté une déclinaison de la procédure présentée par Novikov et al. [NROV14] pour le calcul des marginales d'un modèle SBM. Ensuite, nous

avons présenté une procédure basée sur l'organisation des calculs de ces marginales sur un arbre pour le calcul des marginales d'un SBM et qui peut se généraliser pour n'importe quel modèle graphique avec des liens au plus binaires.
Dans le chapitre 6, nous calculerons des marginales unaires et binaires par l'intermédiaire de l'approche TT avec l'organisation sur un arbre pour un modèle SBM avec des distances que nous simulerons.
Nous évaluerons cette approche en comparant les résultats obtenus avec des résultats obtenus par d'autres approches plus classiques.

Chapitre 6

Évaluation numérique des calculs des marginales d'un modèle SBM par l'approche TT

Table des matières

6.1 Introduction

L'objectif du chapitre est d'évaluer la qualité du calcul des marginales unaires et binaires par l'approche Tenseur Train (TT) présentée dans le chapitre 5. Pour cela, nous la comparons avec trois autres méthodes présentes dans la littérature.

- Premièrement, quand n est petit, nous la comparons avec la méthode de calcul exacte par énumération. Cette dernière consiste à construire le tenseur de la loi jointe puis faire le calcul des marginales par sommation.
  Cette méthode est exacte mais, elle nécessite le stockage d'un tenseur dont un élément est la loi jointe de Z sachant D pour un état de Z : ce tenseur contient Q^n éléments. De plus, le nombre de sommes est exponentiel avec l'ordre du tenseur.
- Ensuite, nous comparons notre approche à une méthode de type Méthode de Monte Carlo par chaîne de Markov [And01] (MCMC) qui sont des méthodes réputées pour leur forte précision mais, qui restent très lentes à mettre en œuvre. Parmi ces méthodes, nous avons choisi l'échantillonneur de Gibbs [RC04], qui est simple à mettre en œuvre. Cette méthode est utilisée dans l'algorithme MCEM.
- Enfin, nous comparons notre approche à une méthode variationnelle basée sur l'approximation en champ moyen [DPR08] de la loi de Z sachant D : c'est une méthode basée sur une hypothèse d'indépendance. Donc, les marginales binaires sont obtenues comme le produit des unaires. Cette méthode est très rapide vu qu'elle repose sur la résolution d'une équation du point fixe mais, peut être peu précise certaines fois. Cette méthode est utilisée dans l'algorithme VEM pour l'estimation des paramètres d'un SBM.

Dans le graphique de la figure 6.1, nous présentons le comportement de chaque méthode en termes de précision et de temps de calcul.

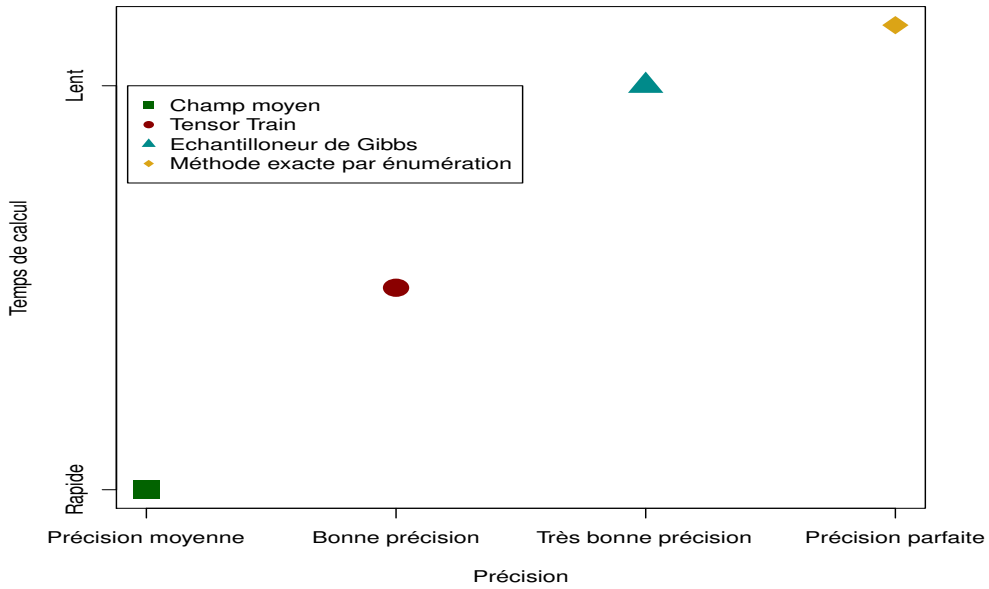


FIGURE 6.1 – Compromis précision/temps de calcul pour les quatre méthodes testées pour l'inférence de la loi de Z sachant D

Afin d'évaluer l'approche TT, nous avons utilisé des matrices de distances simulées à partir de six structures de distances que nous présenterons ultérieurement. Cela a pour but de nous permettre de valider l'approche TT de manière générique.
Ensuite, pour chaque structure :

1. nous avons calculé les marginales unaires et binaires pour les quatre méthodes de calculs (la méthode exacte par énumération, l'échantillonneur de Gibbs, TT et le Champ moyen) pour un nombre d'individus $n = 9$ (qui est la limite à laquelle la méthode du calcul exacte par énumération peut être développée). Puis nous les avons comparé entre eux.
2. nous avons calculé les marginales unaires et binaires pour trois méthodes de calculs (l'échantillonneur de Gibbs, TT et le Champ moyen) pour $n = 18$: la méthode de référence est maintenant l'échantillonneur de Gibbs à laquelle nous avons comparé TT et le champ moyen.

Dans ce qui suit, nous présenterons les quatre méthodes pour le calcul des marginales dans la sous-section 6.2, les structures de distances simulées dans la sous-section 6.3, le protocole pour le calcul des marginales sur les six structures dans la sous-section 6.4, les résultats numériques dans les sous-sections 6.5 et 6.6 et enfin une discussion dans la sous-section 6.7.

6.2 Méthodes de calcul des marginales

Nous commençons par présenter l'expression de la loi jointe dont nous voulons calculer les marginales unaires et binaires, puis nous présentons chacune des trois méthodes autres que "Tenseur Train" pour le calcul des marginales (la méthode via les TT est présentée dans le chapitre 5).

6.2.1 Expression de la loi jointe

La loi jointe de $Z_1, \ldots, Z_n$ conditionnellement aux données D s'exprime en fonction des facteurs unaires (2.2.18) et binaires (2.2.19) présentés dans la section 2.2.1.4 ainsi :

$$\mathbb{P}_\theta(Z_1, \ldots, Z_n \mid D) = \frac{1}{W}\left[\prod_{i=1}^{n}\prod_{j>i}\psi_{i,j}(Z_i, Z_j)\right]\left[\prod_{i=1}^{n}\phi_i(Z_i)\right] \tag{6.2.1}$$

Avec,

$$\psi_{ij} \in \mathbb{R}^{Q\times Q}, \qquad \phi_i \in \mathbb{R}^Q \qquad Z_i, Z_j \in [\![1, Q]\!] \tag{6.2.2}$$

Facteurs Nous exprimons les facteurs binaires et unaires de la loi de Z sachant D :

$$\psi_{i,j}(z_i, z_j) = \sum_{q=1}^{Q}\sum_{q'=1}^{Q}\frac{\lambda_{q,q'}^{D(i,j)}}{D(i,j)!}\exp(-\lambda_{q,q'})z_{i,q} \times z_{j,q'} \tag{6.2.3}$$

$$\phi_i(z_i) = \prod_{q=1}^{Q}\alpha_q^{z_{i,q}} \tag{6.2.4}$$

Pour l'implémentation des calculs, il peut être utile d'assembler les facteurs binaires et unaires dans des nouveaux facteurs binaires de cette manière :

$$\left|\begin{array}{lcll} \psi'_{i,i+1}(z_i, z_{i+1}) & = & \psi_{i,i+1}(z_i, z_{i+1})\phi_i(z_i) & \forall 1 \leq i \leq n-2 \\ \psi'_{i,j}(z_i, z_j) & = & \psi_{i,j}(z_i, z_j) \quad \text{si} \quad j \neq i+1 & \\ \psi'_{n-1,n}(z_{n-1}, z_n) & = & \psi_{n-1,n}(z_{n-1}, z_n)\phi_{n-1}(z_{n-1})\phi_n(z_n) & \end{array}\right. \tag{6.2.5}$$

6.2.2 Méthode exacte par énumération

Le calcul par la méthode exacte par énumération se fait de la manière suivante :

Construction du tenseur de la loi jointe

- Nous créons un tenseur $\mathbf{\Psi} \in (\mathbb{R}^Q)^{\otimes n}$ d'ordre n où nous stockons la loi jointe.
- Puis, $\forall z_1, z_2, \ldots, z_n = 1, \ldots, Q$

$$\mathbf{\Psi}[z_1, z_2, \ldots, z_n] = \frac{1}{W}\left[\prod_{i=1}^{n}\prod_{j>i}\psi_{i,j}(z_i, z_j)\right]\left[\prod_{i=1}^{n}\phi_i(z_i)\right] \tag{6.2.6}$$

Cette méthode consiste à stocker le tenseur complet de la loi jointe dans un ordinateur.

Calcul des marginales Dans ce cas, le calcul des marginales découle de la définition :

- Marginales unaires : $\forall i \in [\![1, n]\!], q \in [\![1, Q]\!]$

$$p_i(Z_i = q) = \sum_{z_1,\ldots,z_{i-1},z_{i+1},\ldots,z_n \in [\![1,Q[\![^{n-1}} \mathbf{\Psi}[z_1, \ldots, z_{i-1}, z_i = q, z_{i+1}, \ldots, z_n] \tag{6.2.7}$$

- Marginales binaires : $\forall i, j \in [\![1, n]\!]^2, q, q' \in [\![1, Q]\!]^2$

$$p_{i,j}(Z_i = q, Z_j = q') = \sum_{z_l \in [\![1,Q]\!], \forall l \in [\![1,n]\!]\setminus\{i,j\}} \mathbf{\Psi}[z_1, \ldots, z_{i-1}, z_i = q, z_{i+1}, \ldots, z_{j-1}, z_j = q', z_{j+1}, \ldots, z_n] \tag{6.2.8}$$

L'avantage de cette méthode est qu'on ne fait pas d'approximation, le calcul est exact. L'inconvénient est qu'elle nécessite le stockage de Q^n éléments, ce qui pose une limite de stockage. De plus, cette méthode nécessite un nombre exponentiel d'additions.

6.2.3 Échantillonneur de Gibbs

Dans cette section, nous commençons par expliquer le principe de l'échantillonneur de Gibbs, sa mise en œuvre et le pré-traitement nécessaire pour son implémention.

6.2.3.1 Principe de l'échantillonneur de Gibbs pour le calcul des marginales

L'échantillonneur de Gibbs [RC04] est un algorithme général de type Monte-Carlo par chaînes de Markov [And01] (MCMC) qui permet de générer des séquences d'observations très proches d'une réalisation d'une loi multivariée difficile à échantillonner directement. Quand la simulation est répétée un grand nombre de fois, cela permet une estimation des marginales.
Le principe est que, étant donné une loi multivariée, il est facile d'échantillonner à partir d'une loi conditionnelle (une variable conditionnellement à toutes les autres). Afin de produire un échantillon indexé par exemple par $\ell \in [\![1, L]\!]$, soit $z^{(\ell)} = \left(z_1^{(\ell)}, \ldots, z_n^{(\ell)}\right)$, nous partons d'un $z^{(\ell)}$ initial. Ensuite, nous mettons à jour successivement sur $i \in [\![1, n]\!]$ et dans l'ordre chacune des composantes $z_i^{(\ell)}$, les autres étant fixées.
Nous répétons en boucle N_{iter} fois ce processus. Lorsque $N_{\text{iter}} \to \infty$, alors le $z^{(\ell)}$ obtenu est une réalisation de la loi $P_\theta(Z \mid D)$, loi jointe de Z sachant D. L'idée pour le calcul des marginales est de produire un grand nombre L d'échantillons qui sont chacun une réalisation approchée de la loi jointe à échantillonner, ici $P_\theta(Z \mid D)$. Des statistiques élémentaires sur les L réalisations permettent alors de calculer les marginales souhaitées selon cette formule :

$$P_\theta(Z_{i,q} = 1 \mid D) \approx \frac{1}{L} \sum_{\ell=1}^{L} \mathbb{1}(z_{i,q}^{(\ell)} = 1) \tag{6.2.9}$$

$$P_\theta(Z_{i,q} = 1, Z_{j,q'} = 1 \mid D) \approx \frac{1}{L} \sum_{\ell=1}^{L} \mathbb{1}(z_{i,q}^{(\ell)} = 1, z_{j,q'}^{(\ell)} = 1) \tag{6.2.10}$$

avec,

$$\begin{cases} z_{i,q}^{(\ell)} = \mathbb{1}(i \in q) \\ \sum_{q=1}^{Q} z_{i,q}^{(\ell)} = 1 \end{cases}$$

Il y a donc deux paramètres qui contrôlent la qualité du résultat :

1. N_{iter} qui contrôle la qualité de chaque réalisation de la loi jointe $P_\theta(Z \mid D)$.
2. L qui contrôle la qualité de l'estimation des marginales.

6.2.3.2 Préparation

Loi conditionnelle : L'approche via l'échantillonneur de Gibbs repose sur la mise à jour séquentielle de chacun des z_i connaissant sa loi conditionnellement à tous les autres z_j. Cette loi s'écrit, pour z_i :

$$P_\theta(Z_i = z_i | \{Z_j = z_j\}_{j \neq i}, D) = \frac{\phi_i(z_i) \prod_{j \neq i} \psi_{i,j}(z_i, z_j)}{\sum_{u=1}^{Q} \phi_i(u) \prod_{j \neq i} \psi_{i,j}(u, z_j)} \tag{6.2.11}$$

Cela suppose de calculer et de stocker les facteurs unaires $\phi_i(z_i)$ et binaires $\psi_{i,j}(z_i, z_j)$.

Initialisation : Nous souhaitons simuler L échantillons.
Pour chaque échantillon $l \in [\![1, L]\!]$, nous prenons un point de départ $z^{(\ell)} = (z_1^{(\ell)}, \ldots, z_n^{(\ell)})$ où nous générons la classe initiale $z_i^{(\ell)}$ de $i \in [\![1, n]\!]$ selon une loi uniforme.

6.2.3.3 Mise en œuvre

L'échantillonneur de Gibbs se fait en deux étapes : une initialisation dans laquelle nous simulons $z^{(\ell)}$ initial et une étape de N_{iter} mises à jour successives de chacune des coordonnées de $z^{(\ell)}$. Le but est d'aller jusqu'à N_{iter} mises à jour de $z^{(\ell)}$ pour que le dernier échantillon produit soit une réalisation fiable de la loi $P_\theta(Z \mid D)$.
La procédure se déroule ainsi :

- **Initialisation** Choisir une valeur initiale $z^{init} = (z_1 \ldots z_n)$.

  Nous attribuons :

  $$z^{old} \leftarrow z^{init}$$

- **Mises à jour successives :** Supposons que l'on commence l'itération t. Nous notons $z^t = z^{old}$ et $z^{t+1} = z^{new}$.
  Une itération de Gibbs consiste à prendre z^{old} en entrée et mettre à jour successivement les états z_i des n nœuds pour retourner z^{new} quand l'ensemble des nœuds ont été mis à jour. La séquence des opérations successives se déroule ainsi :
  - pour $i = 1$, nous simulons z_1^{new} selon la loi définie par $P_\theta(Z_1 = z_1 | \{Z_j^{old} = z^{old}\}_{j \neq 1}, D)$
  - pour $i = 2$, nous simulons z_2^{new} selon la loi définie par $P_\theta(Z_2 = z_2 | Z_1 = z_1^{new}, Z_3 = z_3^{old}, \ldots, Z_n = z_n^{old}, D)$

$$\vdots$$

— pour $i \in [\![1,n]\!]$, nous simulons z_i^{new} selon la loi définie par $P_\theta(Z_i = z_i | Z_1 = z_1^{new}, \dots, Z_{i-1} = z_{i-1}^{new}, Z_{i+1} = z_{i+1}^{old}, \dots Z_n = z_n^{old}, D)$

$$\vdots$$

— pour $i = n$, nous simulons z_n^{new} selon la loi définie par $P_\theta(Z_n = z_n | Z_1 = z_1^{new}, \dots, Z_{n-1} = z_{n-1}^{new}, D)$

— $z^{old} \leftarrow z^{new}$

- Nous refaisons cette simulation de manière séquentielle N_{iter} fois
- Le dernier z^{old} obtenu est la réalisation de la loi $P_\theta(Z|D)$.

Nous lançons L échantillonneurs de Gibbs et nous obtenons L réalisations de Z : $z^{(1)}, z^{(2)}, \dots, z^{(\ell)}$ à partir desquels nous calculons les marginales unaires et binaires selon les formules (6.2.9) et (6.2.10).

6.2.4 Champ moyen

L'approximation basée sur l'approximation en champ moyen de la loi de Z sachant D est une méthode variationnelle développée par Daudin & al. en 2008 [DPR08] dans le cadre de l'algorithme VEM pour l'estimation des paramètres d'un SBM.

6.2.4.1 Principe du champ moyen

Dans le cadre de cette approche, nous souhaitons approcher $\mathbb{P}_\theta(Z \mid D)$ par une distribution $\mathbb{Q}_\theta(Z \mid D)$ plus simple à calculer. Celle-ci doit vérifier les propriétés suivantes :

- P_1 : $\mathbb{Q}_\theta(Z \mid D) = \prod_{i=1}^n q_\theta^i(Z_i \mid D)$
- P_2 : $\mathbb{Q}_\theta$ minimise la divergence de Kullback-Leibler.

Définition 1. *Divergence de Kullback-Leibler [AS66] entre une distribution $\mathbb{Q}$ et une distribution $\mathbb{P}$ s'exprime :*

$$KL(\mathbb{Q}|\mathbb{P}) = \mathbb{E}_{\mathbb{Q}}\left[\log\left(\frac{\mathbb{Q}(Z)}{\mathbb{P}(Z)}\right)\right] \tag{6.2.12}$$

$$KL(\mathbb{Q}|\mathbb{P}) = \sum_z \mathbb{Q}(z)\log\left(\frac{\mathbb{Q}(z)}{\mathbb{P}(z)}\right) = J(\mathbb{Q}(Z)) - \mathbb{E}_{\mathbb{Q}}[\log(\mathbb{P}(Z))] \tag{6.2.13}$$

avec $J(\mathbb{Q}(Z)) = \mathbb{E}_{\mathbb{Q}}[\log(\mathbb{Q}(Z \mid D))]$.

6.2.4.2 Mise en œuvre du champ moyen

Pour cela, nous choisirons la distribution $\mathbb{Q}_\theta$ de cette manière :

$$\mathbb{Q}_\theta = \arg\max_{\mathbb{Q}}(KL(\mathbb{Q} \mid \mathbb{P}_\theta(. \mid D)))$$

Dans le cas optimal, sans contrainte sur la forme de $\mathbb{Q}_\theta$, nous aurons $\mathbb{P}_\theta(Z \mid D) = \mathbb{Q}_\theta(Z \mid D)$ et donc une divergence de Kullback-Leibler nulle.
Rappelons la propriété P_1 :

$$\mathbb{Q}_\theta(Z|D) = \prod_{i=1}^n q_\theta^i(Z_i \mid D)$$

On notera $\tau_{i,q} = q_\theta^i(q \mid D)$. Donc :

$$\forall i \in [\![1,n]\!], \qquad \sum_{q=1}^Q \tau_{i,q} - 1 = 0 \tag{6.2.14}$$

On développe $J(\mathbb{Q}_\theta(Z))$:

$$J(\mathbb{Q}_\theta(Z)) = \mathbb{E}_{\mathbb{Q}_\theta}[\log(\mathbb{Q}_\theta(Z \mid D))] = \mathbb{E}_{\mathbb{Q}_\theta}\left[\sum_{i=n}^{n} \log(q_\theta^i(z_i \mid D))\right]$$

$$= \sum_{i=1}^{n} \mathbb{E}_{\mathbb{Q}_\theta}[\log(q_\theta^i(z_i \mid D))] = \sum_{i=1}^{n}\sum_{q=1}^{Q} \tau_{i,q}\log(\tau_{i,q})$$

Ensuite, nous développons l'espérance de la loi jointe du modèle :

$$\mathbb{E}_{\mathbb{Q}_\theta}[\log(\mathbb{P}_\theta(Z \mid D))] = \mathbb{E}_{\mathbb{Q}_\theta}\left[\log\left(\frac{1}{W} \times \prod_{i=1}^{n}\left[\prod_{j>i} \frac{\lambda_{z_i,z_j}^{D_{i,j}}}{D_{i,j}!}\,\mathrm{e}^{-\lambda_{z_i,z_j}} \prod_{q=1}^{Q} \alpha_q^{z_{i,q}}\right]\right)\right] \tag{6.2.15}$$

Nous appliquons la propriété suivante $\log(a \times b) = \log(a) + \log(b)$ sur la formule (6.2.15) et nous développons :

$$\mathbb{E}_{\mathbb{Q}_\theta}[\log(\mathbb{P}_\theta(Z \mid D))] = \mathbb{E}_{\mathbb{Q}_\theta}(\log(\frac{1}{W})) + \sum_{i=1}^{n}\sum_{j>i}\mathbb{E}_{\mathbb{Q}_\theta}\left[\log(\frac{\lambda_{z_i,z_j}}{D_{i,j}!}\,\mathrm{e}^{-\lambda_{z_i,z_j}^{D_{ij}}})\right] + \sum_{q=1}^{Q}\sum_{i=1}^{n}\mathbb{E}_{\mathbb{Q}_\theta}[z_{i,q}\log(\alpha_q)] \tag{6.2.16}$$

Comme W est une constante, alors : $\mathbb{E}_{\mathbb{Q}_\theta}\left[\log(\frac{1}{W})\right] = \log(\frac{1}{W})$.
Ainsi, nous développons la formule de l'espérance pour obtenir :

$$\mathbb{E}_{\mathbb{Q}_\theta}[\log(\mathbb{P}_\theta(Z \mid D))] = \log(\frac{1}{W}) + \sum_{i=1}^{n}\sum_{j>i}\sum_{q=1}^{Q}\sum_{q'=1}^{Q} \tau_{i,q}\tau_{j,q'}\log\left(\frac{\lambda_{q,q'}^{D_{i,j}}}{D_{i,j}!}\,\mathrm{e}^{-\lambda_{q,q'}}\right) + \sum_{q=1}^{Q}\sum_{i=1}^{n}[\tau_{i,q}\log(\alpha_q)] \tag{6.2.17}$$

Ainsi,

$$\begin{aligned} KL(\mathbb{Q}_\theta \mid \mathbb{P}) = &\sum_{i=1}^{n}\sum_{q=1}^{Q}\tau_{i,q}\log(\tau_{i,q}) - \log(\frac{1}{W}) - \sum_{q=1}^{Q}\sum_{i=1}^{n}[\tau_{i,q}\log(\alpha_q)] \\ &- \sum_{i=1}^{n}\sum_{j>i}\sum_{q=1}^{Q}\sum_{q'=1}^{Q}\tau_{i,q}\tau_{j,q'}\log\left(\frac{\lambda_{q,q'}^{D_{i,j}}}{D_{i,j}!}\,\mathrm{e}^{-\lambda_{q,q'}}\right) \end{aligned} \tag{6.2.18}$$

Nous souhaitons minimiser $KL(\mathbb{Q}_\theta \mid \mathbb{P})$ en respectant la contrainte de la formule (6.2.14). Alors, nous allons minimiser $g(\sigma_l, \tau_{l,q}) = KL(\mathbb{Q}_\theta \mid \mathbb{P}) + \sum_{l=1}^{n}\sigma_l(\sum_{q=1}^{Q}\tau_{l,q} - 1)$ par rapport à σ_l et $\tau_{l,q}$ (avec σ_l le multiplicateur de Lagrange).
Afin de maximiser cette fonction, il faudrait trouver le zéro de sa dérivée par rapport à σ_l et $\tau_{l,q}$.
Nous commençons par calculer la dérivée selon σ_l :

$$\frac{\partial g(\sigma_l, \tau_{l,q})}{\partial \sigma_l} = \sum_{q=1}^{Q}\tau_{l,q} - 1$$

Comme $\frac{\partial g(\sigma_l,\tau_{l,q})}{\partial \sigma_l} = 0$, alors

$$\sum_{q=1}^{Q}\tau_{l,q} = 1 \tag{6.2.19}$$

Comme les matrices D et Λ sont symétriques, la dérivée de $g(\sigma_l, \tau_{l,q})$ selon $\tau_{l,q}$ s'exprime :

$$\frac{\partial g(\sigma_l, \tau_{l,q})}{\partial \tau_{l,q}} = \log(\tau_{l,q}) + 1 + \sigma_l - \log(\alpha_q) - \sum_{j=1}^{l-1} \sum_{q'=1}^{Q} \tau_{j,q'} \left[\log \left(\frac{\lambda_{q,q'}^{D_{l,j}}}{D_{l,j}!} \mathrm{e}^{-\lambda_{q,q'}} \right) \right] - \sum_{j>l} \sum_{q'=1}^{Q} \tau_{j,q'} \left[\log \left(\frac{\lambda_{q,q'}^{D_{l,j}}}{D_{l,j}!} \mathrm{e}^{-\lambda_{q,q'}} \right) \right] \tag{6.2.20}$$

Or,

$$\sum_{j=1}^{l-1} \sum_{q'=1}^{Q} \tau_{j,q'} \left[\log \left(\frac{\lambda_{q,q'}^{D_{l,j}}}{D_{l,j}!} \mathrm{e}^{-\lambda_{q,q'}} \right) \right] + \sum_{j>l} \sum_{q'=1}^{Q} \tau_{j,q'} \left[\log \left(\frac{\lambda_{q,q'}^{D_{l,j}}}{D_{l,j}!} \mathrm{e}^{-\lambda_{q,q'}} \right) \right] = \sum_{j \neq l} \sum_{q'=1}^{Q} \tau_{j,q'} \left[\log \left(\frac{\lambda_{q,q'}^{D_{l,j}}}{D_{l,j}!} \mathrm{e}^{-\lambda_{q,q'}} \right) \right]$$

Ainsi,

$$\frac{\partial g(\sigma_l, \tau_{l,q})}{\partial \tau_{l,q}} = \log(\tau_{l,q}) + 1 + \sigma_l - \log(\alpha_q) - \sum_{j \neq l} \sum_{q'=1}^{Q} \tau_{j,q'} \left[\log \left(\frac{\lambda_{q,q'}^{D_{l,j}}}{D_{l,j}!} \mathrm{e}^{-\lambda_{q,q'}} \right) \right]$$

On cherche :

$$\frac{\partial g(\sigma_l, \tau_{l,q})}{\partial \tau_{l,q}} = 0$$

Donc :

$$\tau_{l,q} = \alpha_q \, \mathrm{e}^{\sum_{j \neq l} \sum_{q'=1}^{Q} \tau_{j,q'} \left[\log \left(\frac{\lambda_{q,q'}^{D_{l,j}}}{D_{l,j}!} \mathrm{e}^{-\lambda_{q,q'}} \right) \right]} \mathrm{e}^{-\sigma_l - 1} \tag{6.2.21}$$

Ensuite, nous utilisons la propriété du log :

$$\sum_{j \neq l} \sum_{q'=1}^{Q} \tau_{j,q'} \left[\log \left(\frac{\lambda_{q,q'}^{D_{l,j}}}{D_{l,j}!} \mathrm{e}^{-\lambda_{q,q'}} \right) \right] = \log \left(\prod_{j \neq l} \prod_{q'=1}^{Q} \left[\frac{\lambda_{q,q'}^{D_{l,j}}}{D_{l,j}!} \mathrm{e}^{-\lambda_{q,q'}} \right]^{\tau_{j,q'}} \right)$$

On pose :

$$\mu_{l,q} = \alpha_q \prod_{j \neq l}^{n} \prod_{q'=1}^{Q} \left[\frac{\lambda_{q,q'}^{D_{l,j}}}{D_{l,j}!} \mathrm{e}^{-\lambda_{q,q'}} \right]^{\tau_{j,q'}} \tag{6.2.22}$$

Ainsi $\tau_{l,q}$ est proportionnel à $\mu_{l,q}$ et s'exprime :

$$\tau_{l,q} = \mu_{l,q} \, \mathrm{e}^{-(\sigma_l + 1)} \tag{6.2.23}$$

On combine les deux formules (6.2.19) et (6.2.24) pour obtenir :

$$\tau_{l,q} = \frac{\mu_{l,q}}{\sum_{q=1}^{Q} \mu_{l,q}} \tag{6.2.24}$$

Une manière de trouver le zéro de cette fonction est de procéder par un schéma du point fixe.

Schéma du point fixe et dynamique des quantités d'intérêt

L'équation de point fixe est la suivante :

$$\tau_{i,q}^{t+1} = \frac{\mu_{i,q}^{t+1/2}}{\sum_{l=1}^{Q} \mu_{i,l}^{t+1/2}} \tag{6.2.25}$$

La dynamique des quantités d'intérêt s'exprime :

$$\forall 1 \leq i \leq n, \forall 1 \leq q \leq Q, \quad \begin{cases} \tau_{i,q}^{t+1} = \frac{\mu_{i,q}^{t+1/2}}{\sum_{l=1}^{Q} \mu_{i,l}^{t+1/2}} \\ \mu_{i,q}^{t+1/2} = \alpha_q \prod_{j \neq i} \prod_{q'=1}^{Q} \left[\frac{\lambda_{q,q'}^{D(i,j)}}{D(i,j)!} \exp -\lambda_{q,q'} \right]^{\tau_{j,q'}^{t}} \end{cases} \tag{6.2.26}$$

On initialise le schéma du point fixe de cette manière :

$$\mu_{i,q}^{0} = \alpha_q$$

Critère d'arrêt : Soit,

$$\forall 1 \leq i \leq n, \forall 1 \leq q \leq Q, \quad \Delta_{i,q} = \frac{|\mu_{i,q}^{t+1} - \mu_{i,q}^{t}|}{|\mu_{i,q}^{t}|}$$

Nous choisissons le critère d'arrêt suivant :

$$\|\Delta\|_\infty < \epsilon_{pf}$$

Après la convergence de ce schéma, nous obtenons les valeurs de $\tau_{i,q}^{*}$ et les marginales unaires s'expriment :

$$q_\theta^i(z_i = q) = \tau_{i,q}^{*} \tag{6.2.27}$$

Enfin, comme pour cette approche où nous faisons une approximation champ moyen, la loi jointe est approchée par le produit des marginales unaires, le calcul des marginales binaires s'obtient comme produit de marginales unaires :

$$\forall i, j \in [1, n]^2, q_\theta^{i,j}(z_i, z_j) = q_\theta^i(z_i) q_\theta^j(z_j) \tag{6.2.28}$$

6.3 Les six structures de réseaux à utiliser pour générer les graphes

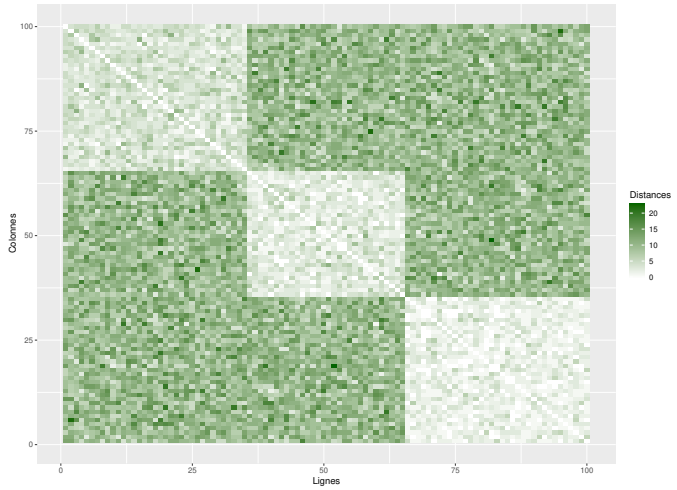
Il existe plusieurs structures de SBM qui permettent de modéliser des systèmes assez différents comme la détection de communautés dans les réseaux sociaux et de composants fonctionnels dans les réseaux d'interaction protéine-protéine qui ont été illustrées par exemple dans l' article [KBKK].En nous basant sur la littérature [KBKK, FYZS18, FB19, BE99], nous avons choisi les six structures suivantes pour simuler nos matrices des distances car elles correspondent à des structures de lambda bien contrastées. Nous présentons ci dessous les structures de distances utilisées et nous illustrons les matrices de distances simulée pour un exemple $D \in \mathbb{R}^{100\times100}$.

La structure associative [KBKK] les distances classes sont faibles tandis que les distances interclasses sont fortes.

Structure Λ_{asso}

$$\Lambda_{\text{asso}} = \begin{pmatrix} 2 & 10 & 10 \\ 10 & 3 & 10 \\ 10 & 10 & 4 \end{pmatrix}$$

Heatmap d'une matrice simulée

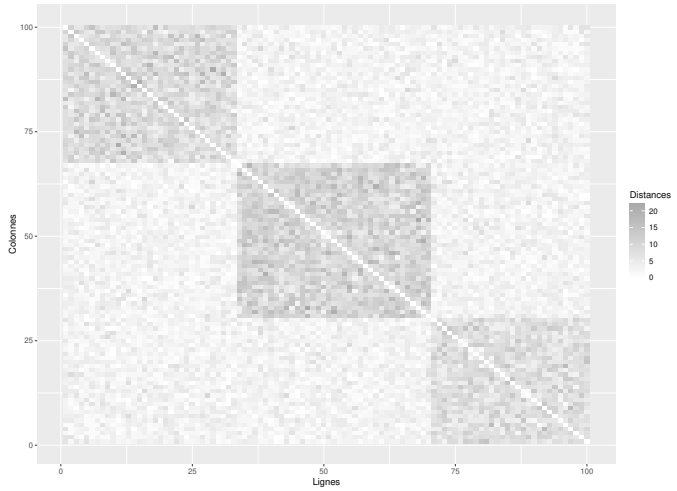


Structure dissociative [KBKK] les distances intra classes sont fortes tandis que les distances interclasses sont faibles.

Structure Λ_{diss}

$$\Lambda_{\text{diss}} = \begin{pmatrix} 8 & 3 & 3 \\ 3 & 10 & 3 \\ 3 & 3 & 9 \end{pmatrix}$$

Heatmap d'une matrice simulée



Structure hierarchique [KBKK] où les distances intra classes sont plus fortes que les distances interclasses. De plus, les distances sont organisées de manière hiérarchique. C'est-à-dire :

Les distances intra classes vérifient :

$$\Lambda_{\text{hier}}[1,1] > \Lambda_{\text{hier}}[2,2] > \Lambda_{\text{hier}}[3,3]$$

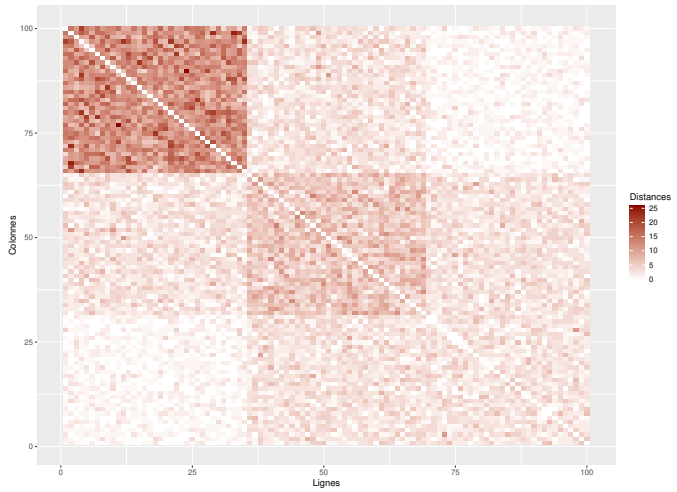
Les distances interclasses vérifient :

$$\Lambda_{\text{hier}}[1,2] \geq \Lambda_{\text{hier}}[2,3] \geq \Lambda_{\text{hier}}[1,3]$$

Structure Λ_{hier}

$$\Lambda_{\text{hier}} = \begin{pmatrix} 12 & 3 & 1 \\ 3 & 6 & 3 \\ 1 & 3 & 3 \end{pmatrix}$$

Heatmap d'une matrice simulée



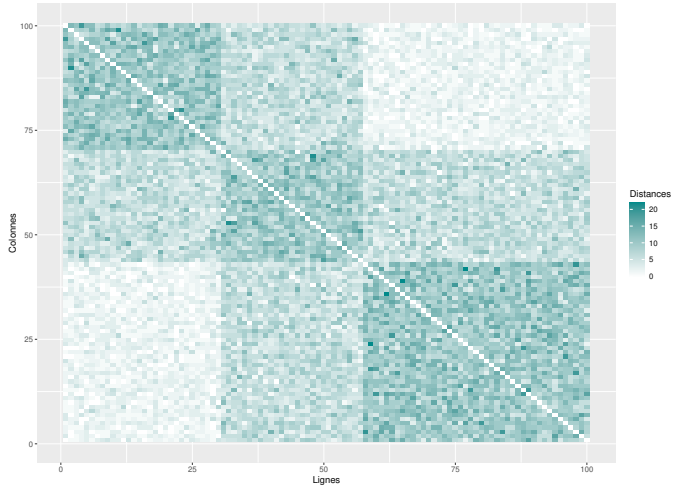
Ordonné [KBKK] Les distances intra classes sont beaucoup plus fortes que les distances interclasses.
De plus, les distances interclasses sont organisés de manière décroissante et vérifient :

$$\Lambda_{\text{hier}}[1,2] \geq \Lambda_{\text{hier}}[2,3] \geq \Lambda_{\text{hier}}[1,3]$$

Structure Λ_{ord}

$$\Lambda_{\text{ord}} = \begin{pmatrix} 10 & 6 & 2 \\ 6 & 10 & 6 \\ 2 & 6 & 10 \end{pmatrix}$$

Heatmap d'une matrice simulée



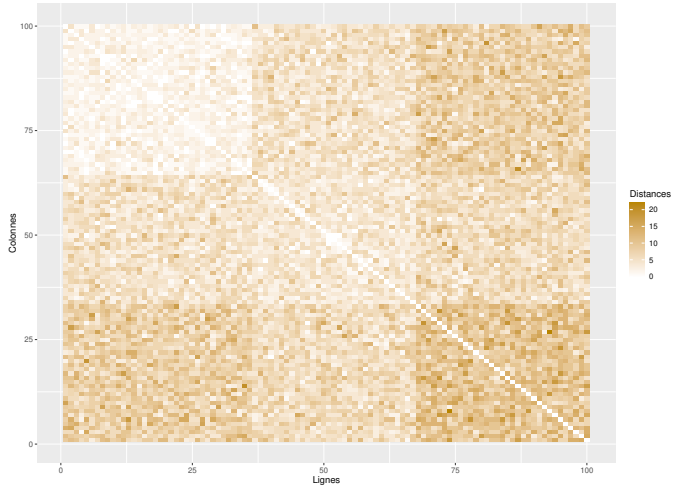
Cores periphery [BE99, FYZS18, FB19], cette structure, suppose qu'il existe deux classes de nœuds, ceux du centre et ceux de la péripherie. Les nœuds adoptent ce comportement :

- Les nœuds du centre qui sont fortement interconnectés et intra connectés. Leurs distances intra et interclasses sont faibles.
- Les nœuds périphériques ne sont pas connectés à d'autres nœuds périphériques. Les nœuds d'une même classe peuvent être fortement connectés entre eux.

Structure Λ_{cp}

$$\Lambda_{\text{cp}} = \begin{pmatrix} 2 & 5 & 8 \\ 5 & 4 & 6 \\ 8 & 6 & 10 \end{pmatrix}$$

Heatmap d'une matrice simulée



Hiérarchique 2 [FB19] Dans la littérature, nous avons trouver une seconde structure hiérarchique que nous appelons hiérarchique 2 afin d'éviter la confusion. Dans cette structure, les distances intra classes sont plus faibles que les distances interclasses. De plus, les distances sont organisées de manière hiérarchique "croissante".
C'est-à-dire :
Les distances intra classes vérifient :

$$\Lambda_{\text{hier}}[3,3] > \Lambda_{\text{hier}}[2,2] > \Lambda_{\text{hier}}[1,1]$$

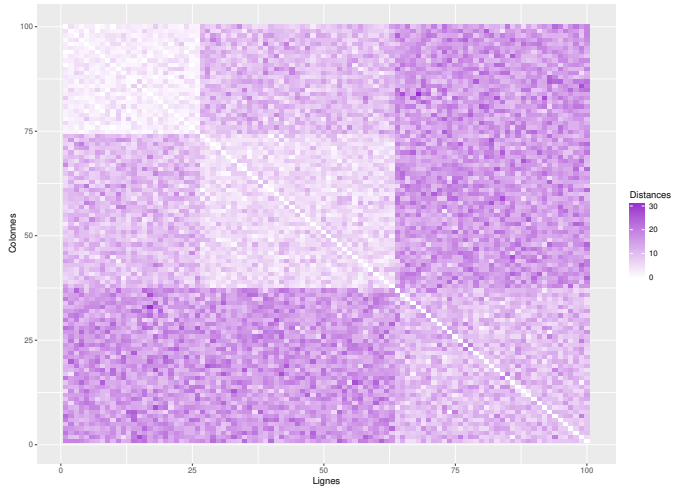
Les distances interclasses vérifient :

$$\Lambda_{\text{hier}}[1,2] \leq \Lambda_{\text{hier}}[2,3] \leq \Lambda_{\text{hier}}[1,3]$$

Structure Λ_{h2}

$$\Lambda_{\text{h2}} = \begin{pmatrix} Th \end{pmatrix}$$

Heatmap d'une matrice simulée



6.4 Protocole pour l'évaluation de l'approche TT

Objectif : Évaluer l'approche TT pour le calcul des marginales unaires et binaires. Pour cela, nous comparons avec les trois méthodes présentées dans la section 6.2.
Dans ce but, nous adopterons la procédure suivante pour chacune parmi les six structures de distances présentées dans la section 6.3 :
Pour une structure de distance donnée :

1. Fixer le nombre de classes $Q = 3$ et $\alpha = \begin{pmatrix} \frac{1}{Q} & \frac{1}{Q} & \frac{1}{Q} \end{pmatrix}$
2. Choisir le nombre d'individus n.
3. Affecter à chaque nœud i une classe (Z_i et q tel que $Z_{i,q} = 1$) selon α.
4. Simuler D connaissant les classes de chaque nœud et la structure de distance choisie.
5. Calculer les facteurs unaires et binaires.
6. Calculer les marginales unaires et binaires pour chaque méthode.
7. Comparer les résultats obtenus par paire de méthode.

Dans les deux sous-sections suivantes, nous présentons la méthode avec laquelle nous simulons les matrices de distances puis les paramètres utilisés pour nos expérimentations.

6.4.1 Simulation des matrices de distances

Pour un nombre d'individus n, un nombre de classes Q, une structure de distances entre classes Λ et un vecteur de proportion d'individus dans les classes α, nous simulons la matrice de distances D de cette manière.
Étape 1 : Simulation des classes des individus :

- Pour chaque individu $i = 1, \dots n$, nous simulons une valeur Z_i qui sera sa classe par l'intermédiaire d'une loi multinomiale de paramètre α.

Étape 2 : Simulation des matrices de distances :

- Pour chaque $i \in [\![1, n]\!]$ et $j \in [\![i, n]\!]$, nous génèrons $D(i, j)$ selon une loi de Poisson de paramètre λ_{Z_i, Z_j}.

6.4.2 Paramètres des méthodes

Voici le paramétrage choisi pour la mise en œuvre des méthodes de calcul des marginales.

- **TT**
  - Nous faisons des décompositions en valeurs singulières complètes des matrices des facteurs.
  - Nous faisons les calculs des produits des TT matrices avec une précision du rounding ϵ égale à 10^{-2} pour $n = 9$ et 10^{-3} pour $n = 18$.
- **Champ moyen :** Nous avons fixé le critère d'arrêt ϵ_{pf} du schéma du point fixe à 10^{-2} pour toutes les expériences.
- **Échantillonneur de Gibbs :** Concernant cette méthode, nous adaptons le nombre d'échantillonneurs de Gibbs (L) et d'itérations (N_{iter}) selon la structure de distance. En effet, nous avons remarqué que pour certaines structures de distances, nous avons besoin d'augmenter le nombre d'itérations N_{iter} pour simuler une réalisation précise de la loi jointe ainsi que le nombre d'itérations L pour améliorer la précision de l'estimation des marginales.
  Dans les tableaux ci-dessous, nous présentons les paramètres choisis.

- $n = 9$

Structure de distance	associative	hiérarchique	ordonnée	dissociative	core-periphérie	hiérarchique 2
Nombre d'échantillons	10000	1000	1000	5000	1000	5000
Nombre d'itérations	1000	500	500	500	500	1000

- $n = 18$

Structure de distance	associative	hiérarchique	ordonnée	dissociative	core-periphérie	hiérarchique 2
Nombre d'échantillons	7000	7000	1000	12000	1000	14000
Nombre d'itérations	1000	1000	500	500	500	500

6.4.3 Visualisation des résultats

Pour la visualisation des différences entre les marginales unaires et binaires pour les quatre méthodes, nous avons choisi d'utiliser les pairplots fournis par la librairie ggplot2 du langage R.
Un pairplot est une matrice 4 par 4 de graphiques :

- **Diagonale** La densité des marginales unaires et binaires pour chaque méthode.
- **Partie triangulaire supérieure :** coefficient de corrélation entre les marginales (unaires ou binaire) pour chaque paire de méthode.
- **Partie triangulaire inférieure :** Le tracé des marginales (unaires ou binaire) d'une méthode en fonction de celle d'une autre méthode. Si les deux méthodes calculent des marginales identiques, ce graphique correspond au tracé de la droite $x = y$.

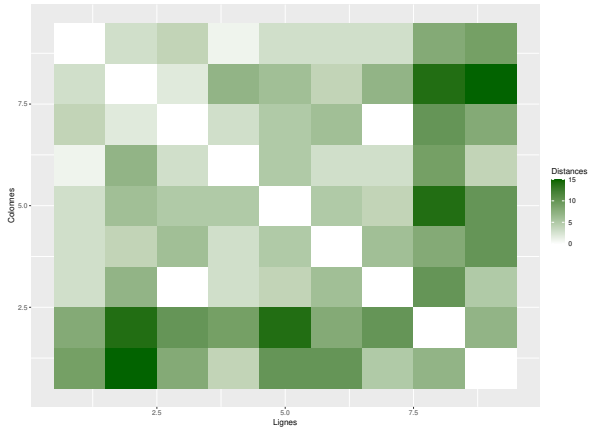
6.4.4 Présentation des résultats

Dans les deux sections suivantes, nous présentons les résultats de comparaisons entre les méthodes via des pairplots entre chaque paire de méthodes à la fois pour les marginales unaires et binaires.
Les résultats seront présentés de cette manière, pour chaque structure des distances, nous présentons les classes simulées, une heatmap de la matrice des distances organisée selon les classes, puis les pairplots des comparaisons des méthodes par paire à la fois pour les marginales unaires à gauche puis les binaires à droite.

6.5 Comparaison pour un nombre d'individus égale à 9

6.5.1 Associative

- **Classes simulées** $\begin{pmatrix} 2 & 2 & 2 & 2 & 1 & 2 & 2 & 0 & 2 \end{pmatrix}$
- **Heatmap de la matrice des distances**



- **Pairplots**

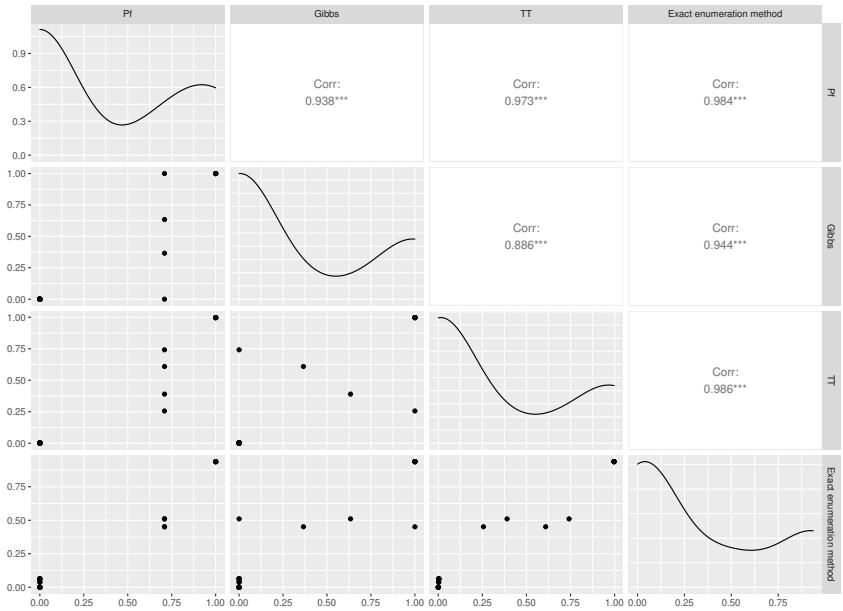


FIGURE 6.2 – Marginales unaires (structure associative (n=9))

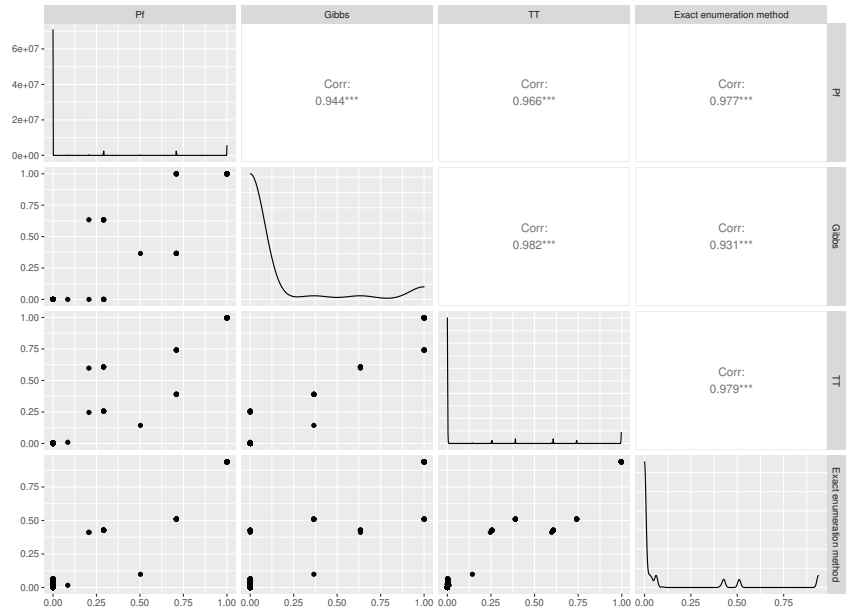
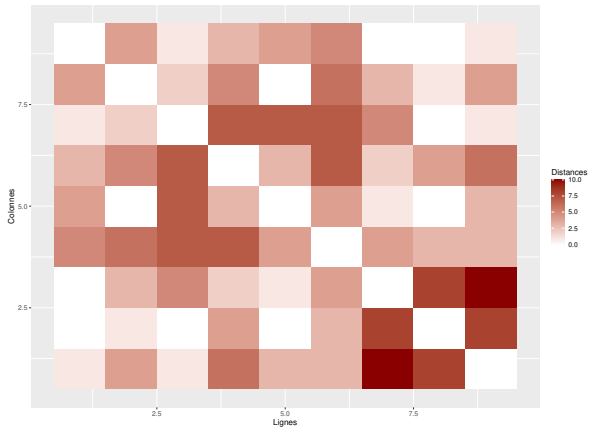


FIGURE 6.3 – Marginales binaires (structure associative (n=9))

6.5.2 Hiérarchique

- **Classes simulées** $\begin{pmatrix} 0 & 1 & 0 & 0 & 2 & 2 & 1 & 1 & 1 \end{pmatrix}$

- **Heatmap de la matrice des distances**



- **Pair plots**

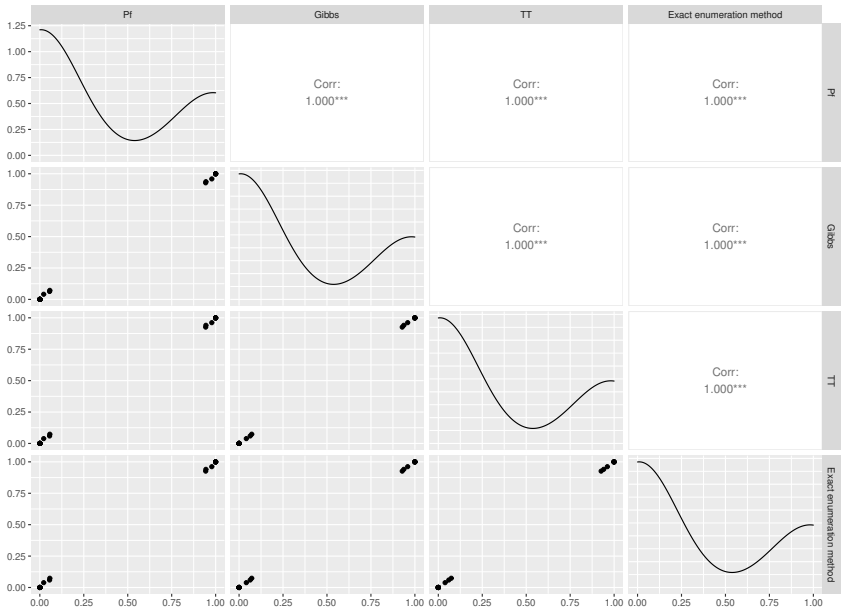


FIGURE 6.4 – Marginales unaires (structure hiérarchique (n=9))

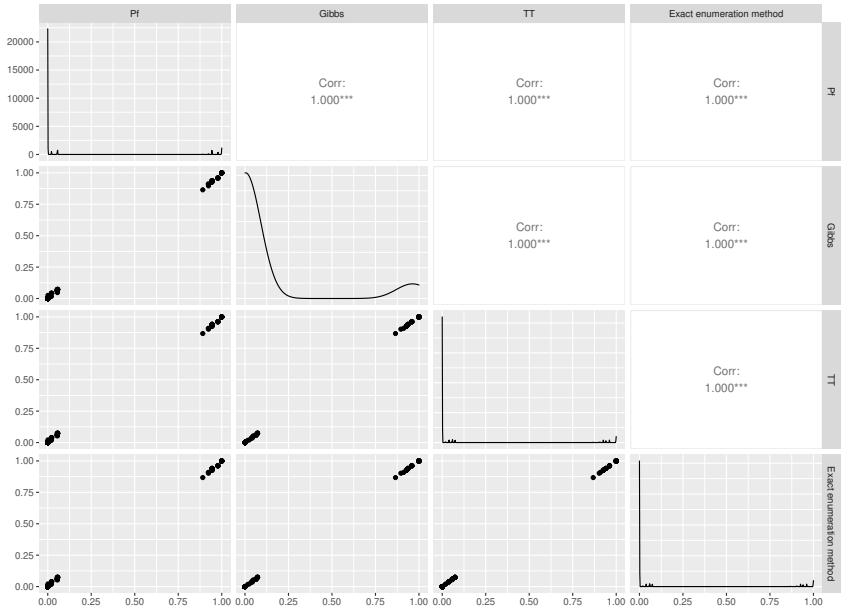
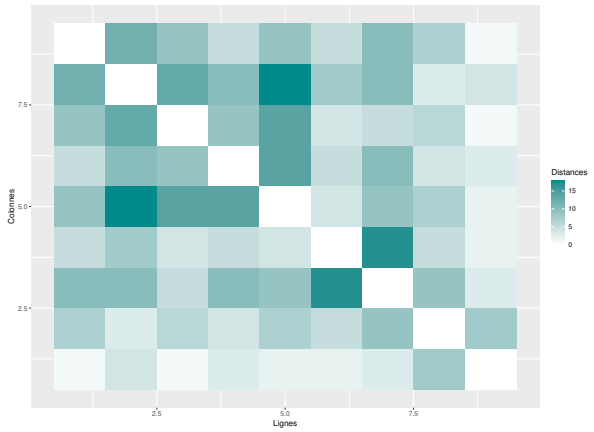


FIGURE 6.5 – Marginales binaires (structure hiérarchique (n=9))

6.5.3 Ordonné

- **Classes simulées** $\begin{pmatrix} 2 & 2 & 1 & 2 & 1 & 2 & 1 & 0 & 2 \end{pmatrix}$
- **Heatmap de la matrice des distances**



- **Pair plots**

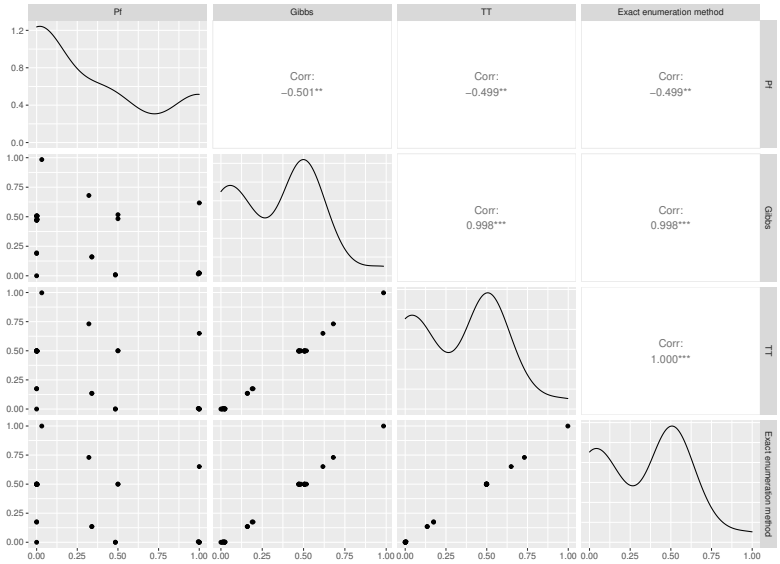


FIGURE 6.6 – Marginales unaires (structure ordonné (n=9))

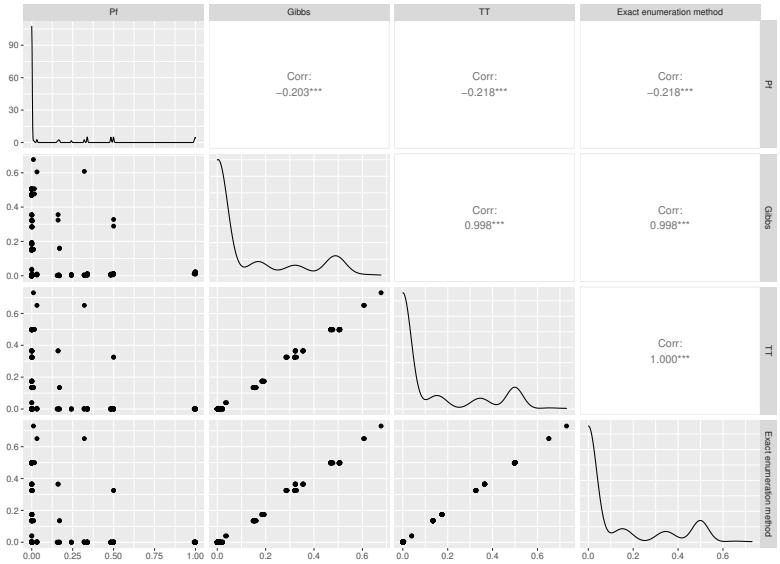
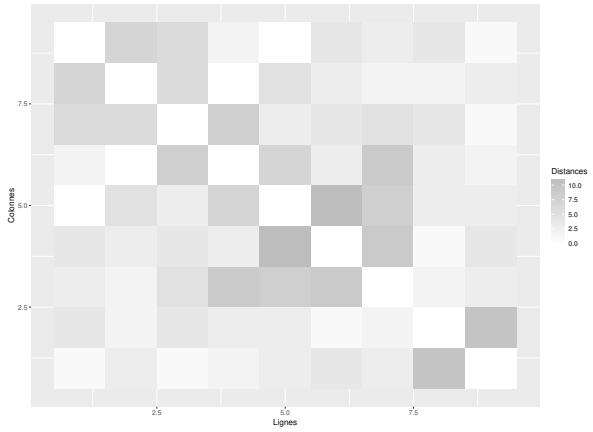


FIGURE 6.7 – Marginales binaires (structure ordonné (n=9))

6.5.4 Dissociative

- **Classes simulées** $\begin{pmatrix} 1 & 1 & 0 & 2 & 2 & 1 & 2 & 0 & 1 \end{pmatrix}$

- **Heatmap de la matrice des distances**



- **Pair plots**

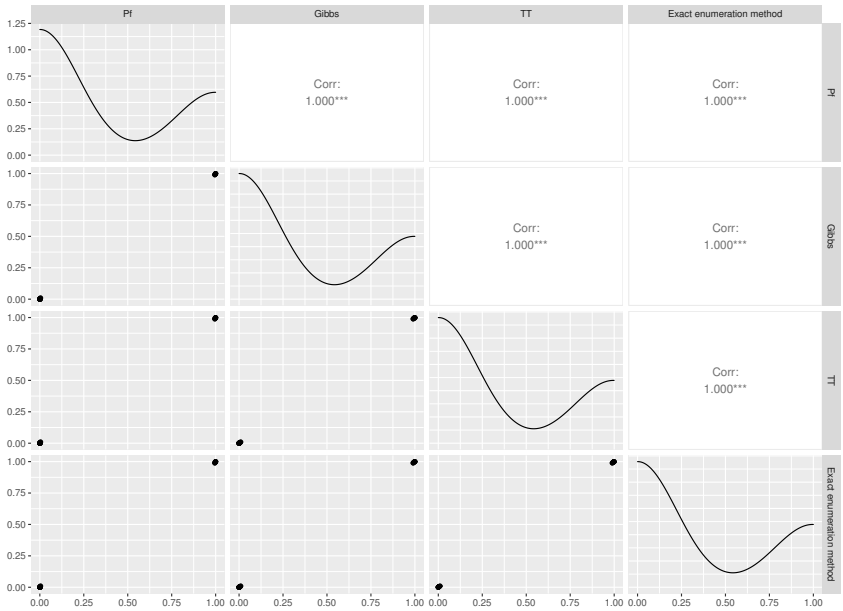


FIGURE 6.8 – Marginales unaires (structure dissociative (n=9))

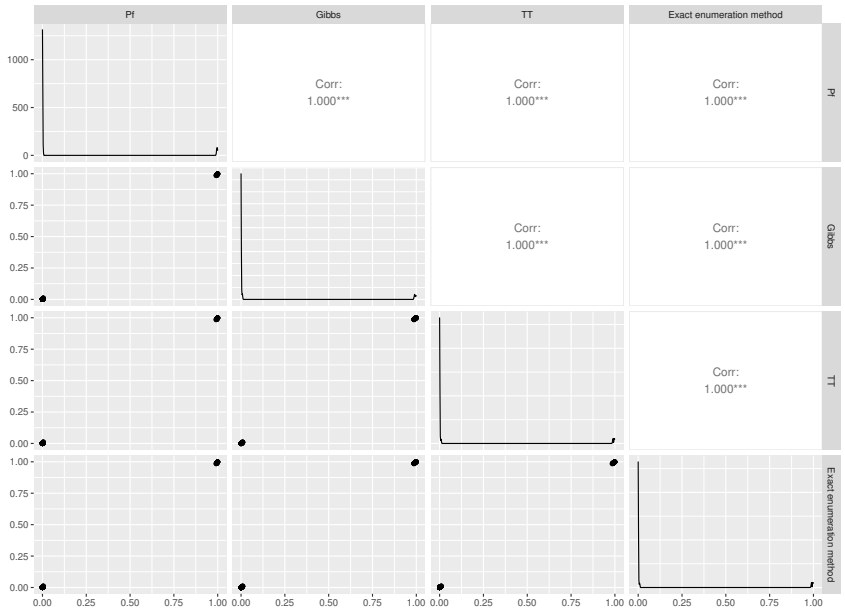
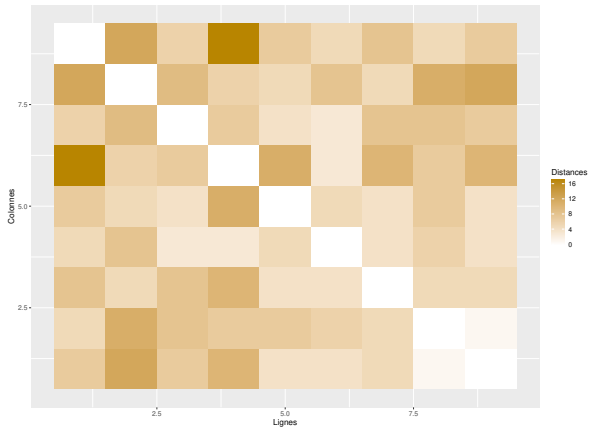


FIGURE 6.9 – Marginales binaires (structure dissociative (n=9))

6.5.5 Cores periphery

- **Classes simulées** $\begin{pmatrix} 2 & 0 & 2 & 2 & 0 & 1 & 2 & 0 & 0 \end{pmatrix}$

- **Heatmap de la matrice des distances**



- **Pair plots**

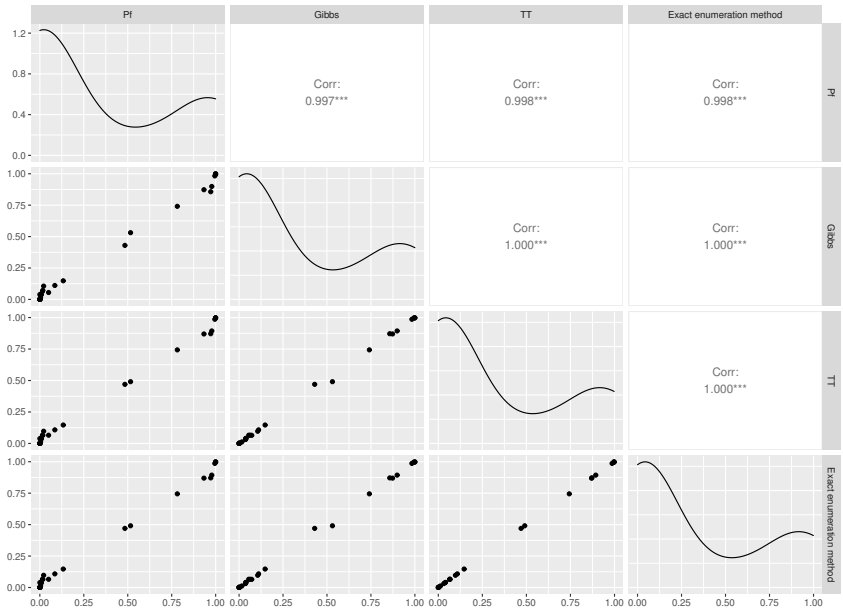


FIGURE 6.10 – Marginales unaires (structure Cores periphery (n=9))

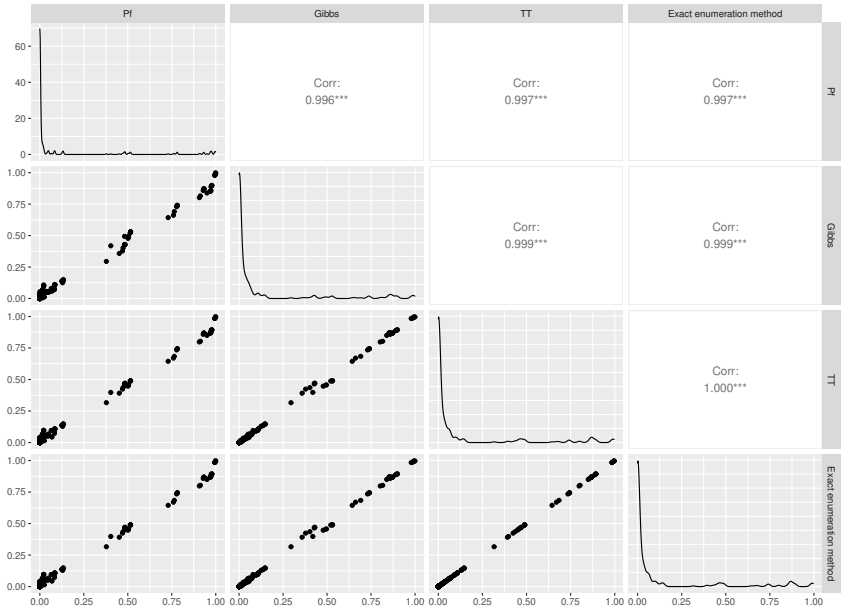
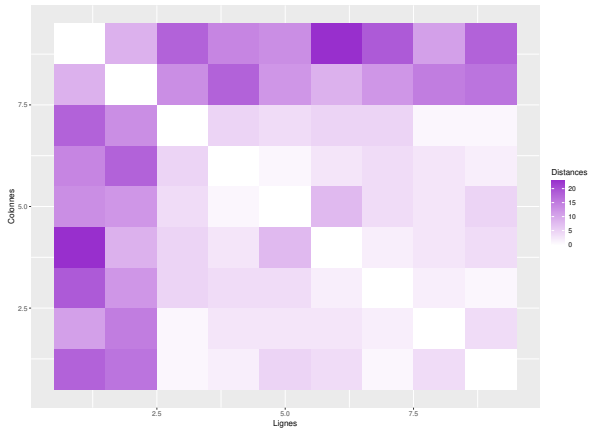


FIGURE 6.11 – Marginales binaires (structure Cores periphery (n=9))

6.5.6 Hiérarchique 2

- **Classes simulées** $\begin{pmatrix} 0 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
- **Heatmap de la matrice des distances**



- **Pair plots**

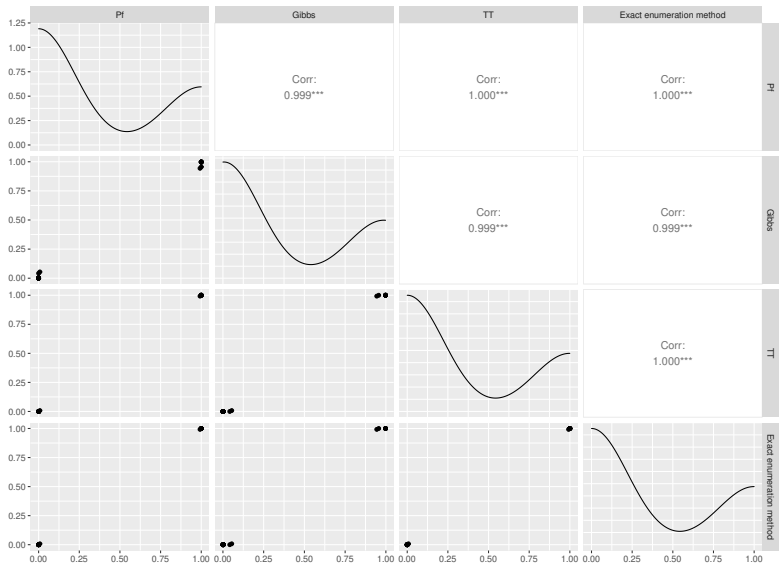


FIGURE 6.12 – Marginales unaires (structure hiérarchique 2 (n=9))

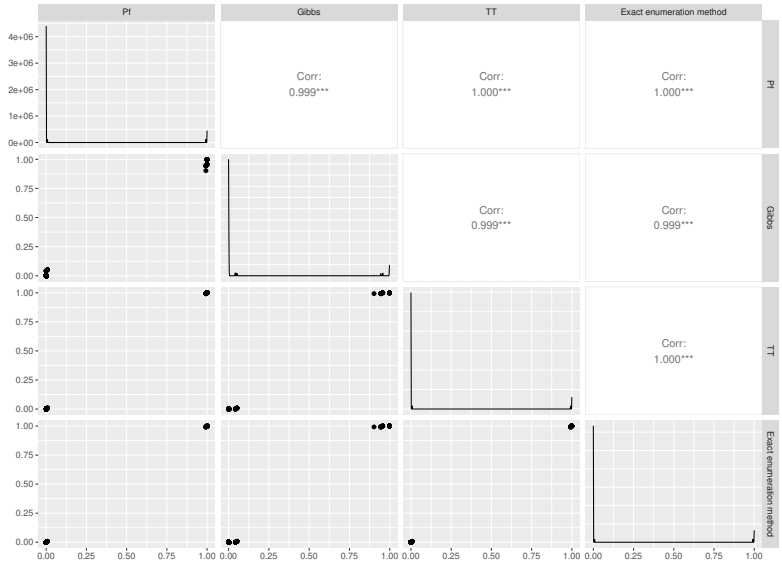


FIGURE 6.13 – Marginales binaires (structure hiérarchique 2 (n=9))

6.5.7 Remarques sur le temps de calcul

Les expériences ont été réalisées sans parallélisation, sur un ordinateur présentant les caractéristiques suivantes : Processeur : Intel(R) Core(TM) i9-9880H CPU @ 2.30GHz ainsi qu'une mémoire totale de 62 go (pour **15 go utilisées**). Voici un tableau récapitulatif des temps des calculs :

Méthode / Temps de calcul (s)	Point fixe	Gibbs	TT
	autour de 0.02	entre 189 et 3608	entre 2.12 et 6

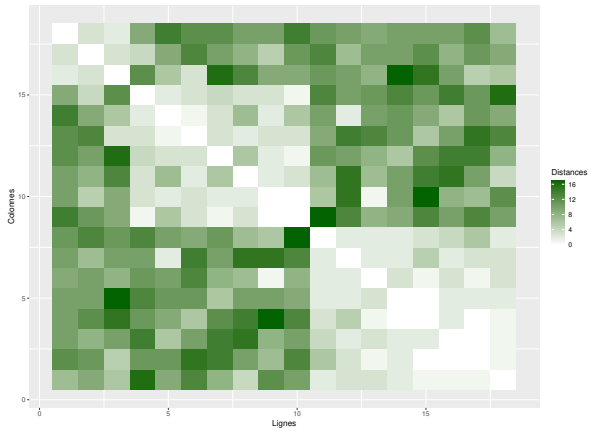
Table 6.1 – Tableau présentant les temps de calcul des marginales unaires et binaires, pour $n = 9$.

6.6 Comparaison pour un nombre d'individus égal à 18

6.6.1 Associative

- **Classes simulées** $\begin{pmatrix} 2 & 1 & 2 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 2 \end{pmatrix}$

- **Heatmap de la matrice des distances**



- **Pair plots**

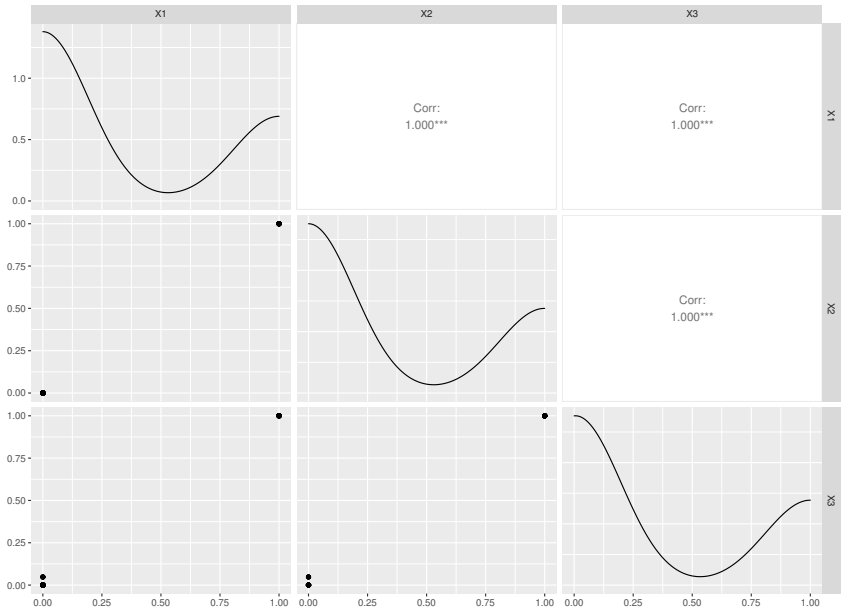


FIGURE 6.14 – Marginales unaires (structure associative (n=18))

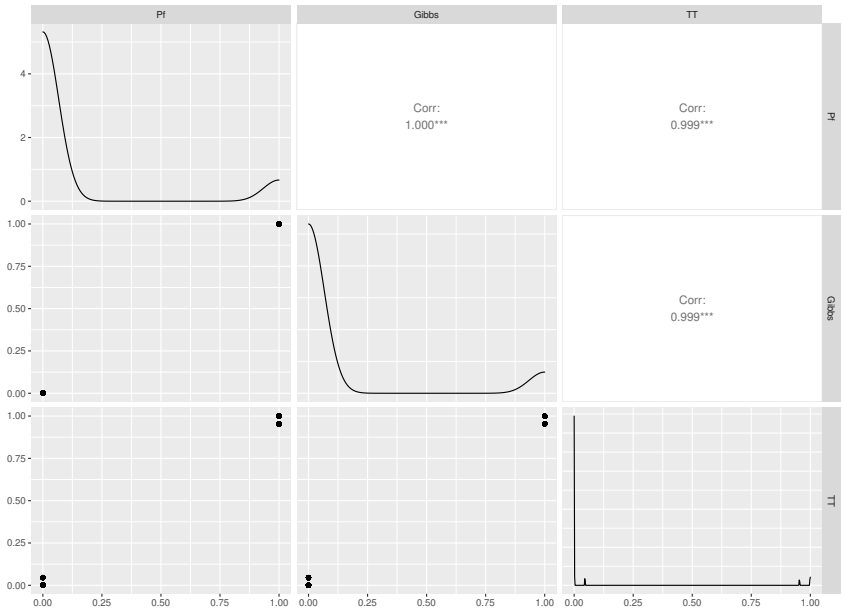
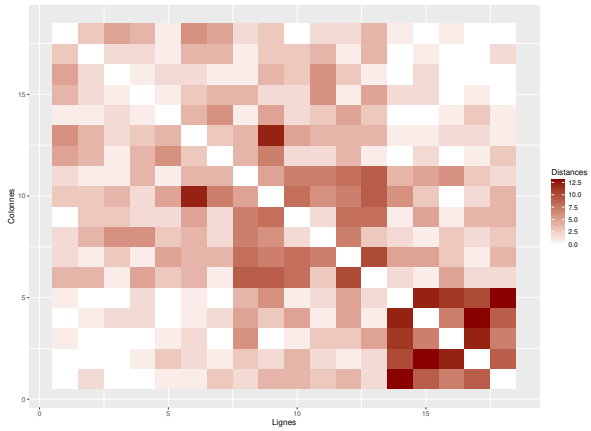


FIGURE 6.15 – Marginales binaires (structure associative (n=18))

6.6.2 Hiérarchique

- **Classes simulées** $\begin{pmatrix} 2 & 0 & 2 & 1 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 1 \end{pmatrix}$

- **Heatmap de la matrice des distances**



- **Pair plots**

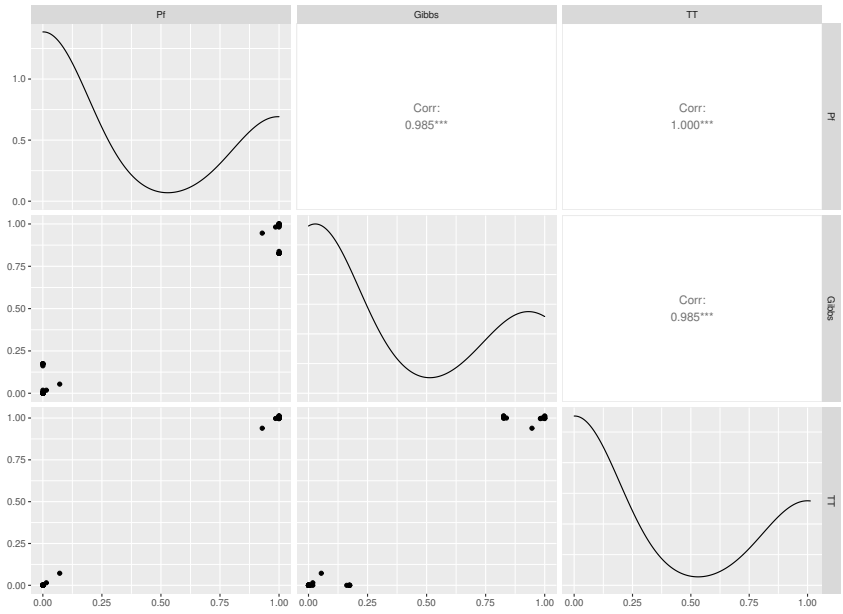


FIGURE 6.16 – Marginales unaires (structure hiérarchique (n=18))

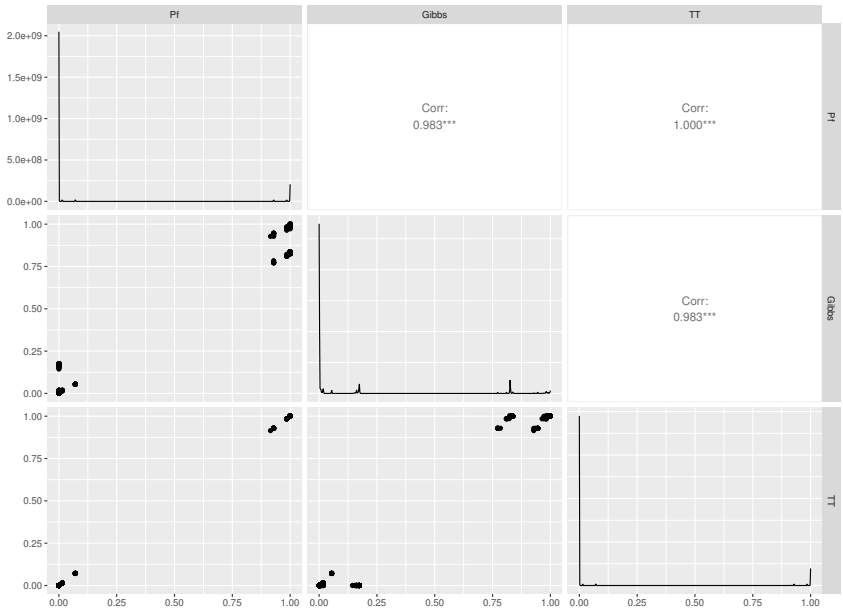
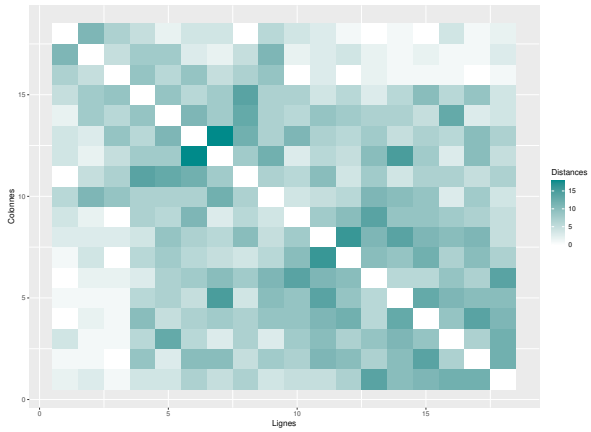


FIGURE 6.17 – Marginales binaires (structure hiérarchique (n=18))

6.6.3 Ordonné

- **Classes simulées** $\begin{pmatrix} 0 & 0 & 1 & 2 & 2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 2 \end{pmatrix}$
- **Heatmap de la matrice des distances**



- **Pair plots**

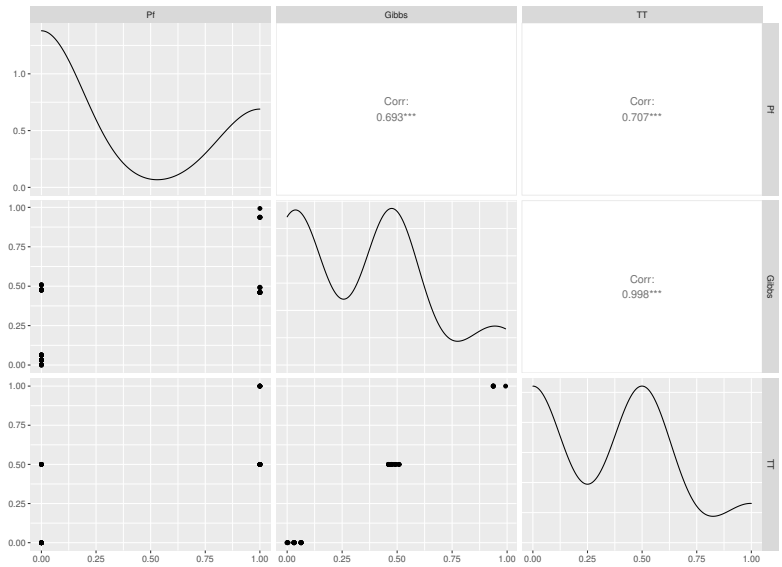


FIGURE 6.18 – Marginales unaires (structure ordonné (n=18))

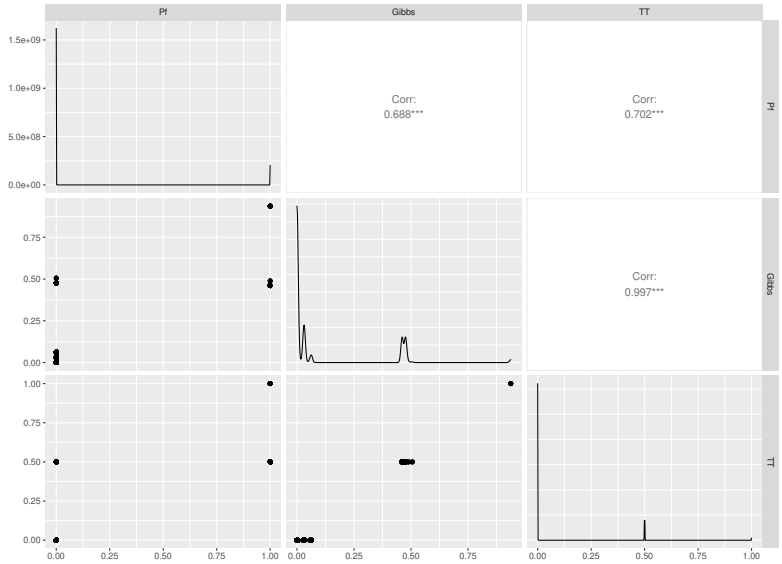
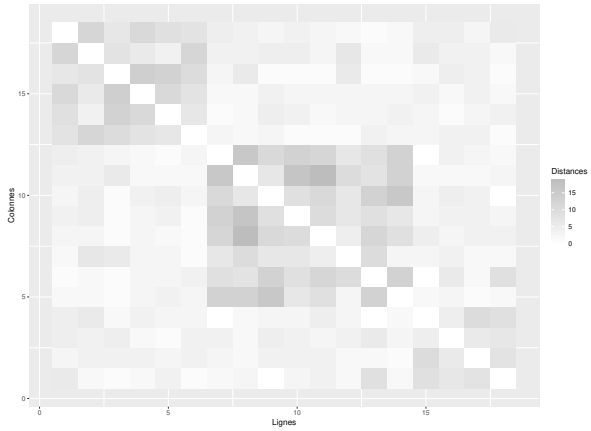


FIGURE 6.19 – Marginales binaires (structure ordonné (n=18))

6.6.4 Dissociative

- **Classes simulées** $\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 2 & 2 & 0 & 1 & 1 & 2 & 2 & 2 & 1 & 1 & 2 & 1 \end{pmatrix}$

- **Heatmap de la matrice des distances**



- **Pair plots**

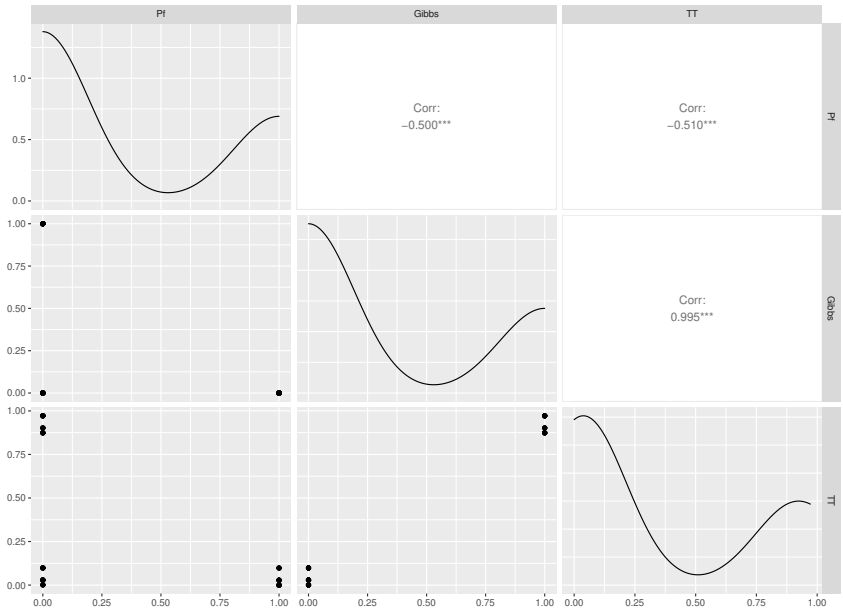


FIGURE 6.20 – Marginales unaires (structure dissociative (n=18))

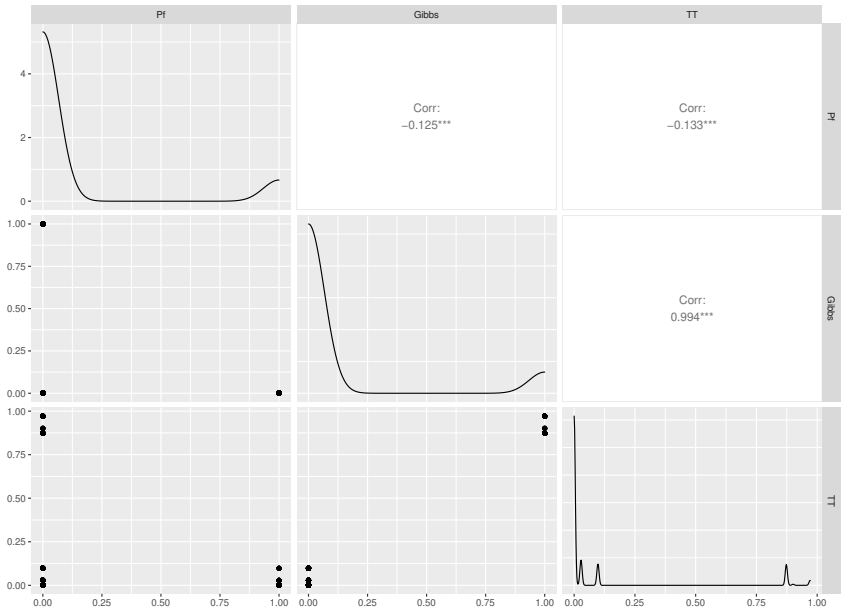
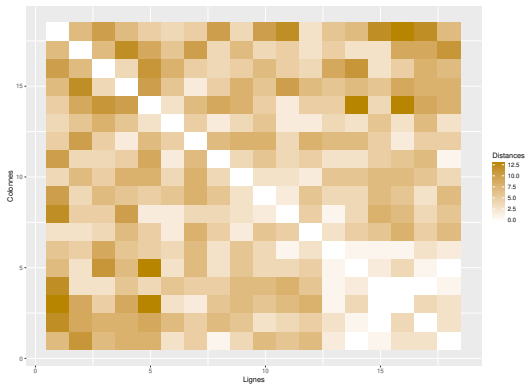


FIGURE 6.21 – Marginales binaires (structure dissociative (n=18))

6.6.5 Cores periphery

- **Classes simulées** $\begin{pmatrix} 2 & 2 & 1 & 2 & 1 & 0 & 0 & 2 & 0 & 1 & 2 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$

- **Heatmap de la matrice des distances**



- **Pair plots**

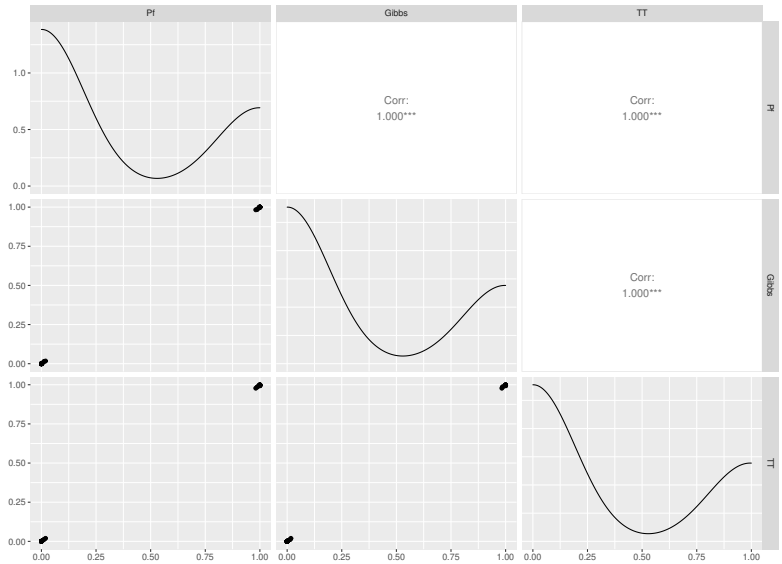


FIGURE 6.22 – Marginales unaires (structure Cores periphery (n=18))

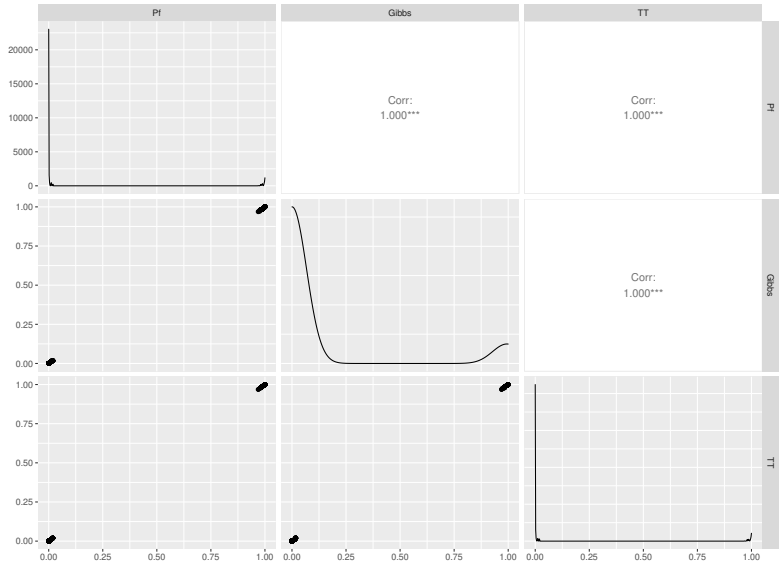
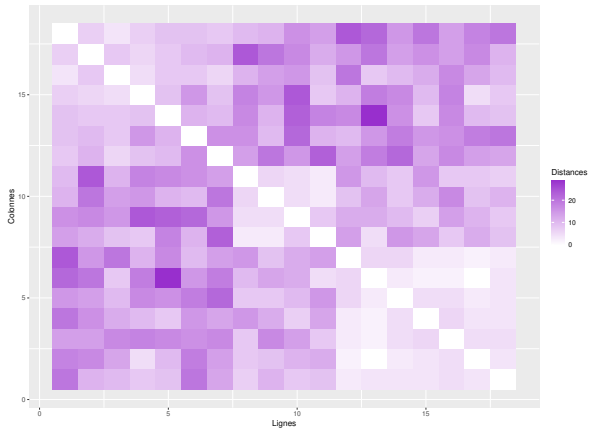


FIGURE 6.23 – Marginales binaires (structure Cores periphery (n=18))

6.6.6 Hiérarchique 2

- **Classes simulées** $\begin{pmatrix} 2 & 1 & 2 & 2 & 1 & 1 & 2 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 2 & 0 & 0,2 \end{pmatrix}$
- **Heatmap de la matrice des distances**



- **Pair plots**

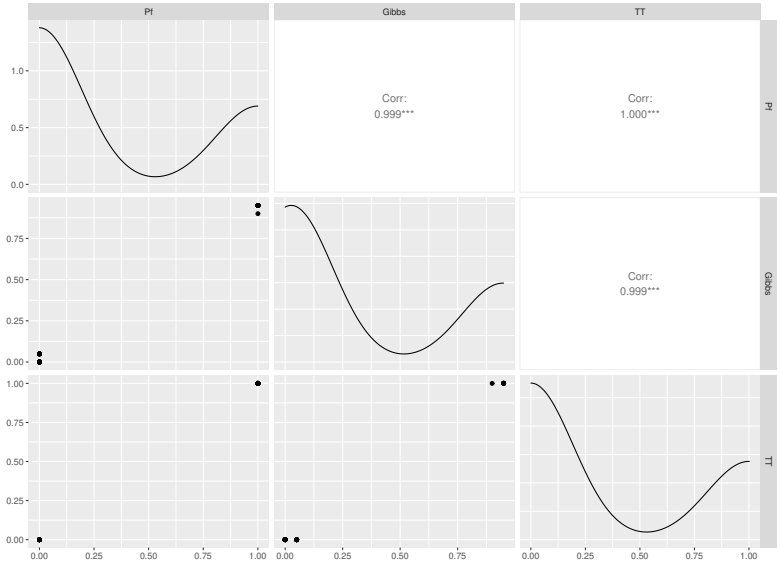


FIGURE 6.24 – Marginales unaires (structure hiérarchique 2 (n=18))

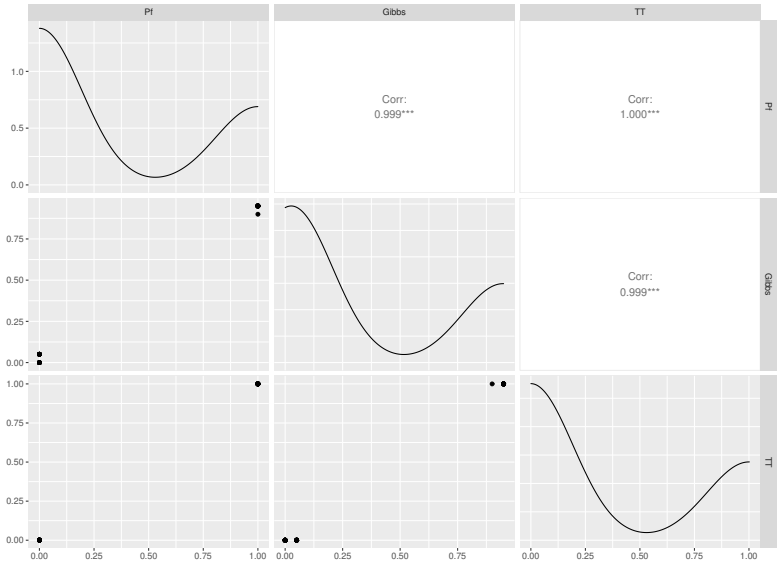


FIGURE 6.25 – Marginales binaires (structure hiérarchique 2 (n=18))

6.6.7 Remarque sur le temps de calcul

Comme pour un nombre d'individus $n = 9$, nous présentons les temps de calcul que chaque méthode a nécessité pour le calcul des marginales pour des SBM avec $n = 18$.

Méthode / Temps de calcul (s)	Point fixe	Gibbs	TT
	Entre 0.4 et 0.6	Entre 2834 et 8139	Entre 50 et 400

TABLE 6.2 – Tableau présentant les temps de calcul des marginales unaires et binaires, pour $n = 18$.

Nous observons dans les tableaux 6.1 et 6.2 que le temps est multiplié par un facteur entre 20 et 30 pour la méthode du Point Fixe, il est multiplié par un facteur de 50 pour l'approche TT. Globalement, le temps de calcul augmente énormément entre les deux familles d'expériences.

6.7 Discussion

Nous commencerons cette section par un résumé des résultats que nous avons présentés durant ce chapitre, ensuite nous nous intéresserons à la cohérence entre les différentes méthodes de calcul des marginales. Enfin, nous nous pencherons sur la non convergence du schéma point du point fixe utilisé dans la méthode du champs moyen.

6.7.1 Résumé des résultats

Nous avons présenté deux expériences où nous avons comparé différentes méthodes de calculs de marginales unaires et binaires d'un SBM sur six structures de distances.
Pour chaque nombre d'individus et chaque structure de distance, nous avons présenté les résultats par l'intermédiaire de pairplots.

- **Comparaisons pour un nombre d'individus $n = 9$:** nous prenons comme référence la méthode exacte par énumération.
  Dans nos résultats, nous montrons une dépendance entre les précisions des méthodes et la structure de distances à partir de laquelle nous avons simulé la matrice de distance. Mais généralement, voici ce que nous pouvons dire pour les six structures de distances :
  - **Comparaison entre Gibbs, TT et la méthode exacte par énumération :** Les coefficients de corrélations entre l'approche de Gibbs (respectivement l'approche TT) et notre référence varient entre 0.944 et 1 (respectivement entre 0.986 et 1) pour les marginales unaires et 0.931 et 1 (respectivement entre 0.979 et 1) pour les binaires. De plus, les scatterplots confirment cette proximité entre ces deux méthodes et notre référence.
    Nous avons en plus établi que l'approche de Gibbs nécessite dans certains cas un nombre d'échantillonneurs L et d'itérations N_{iter} très élevés qui conduisent à des temps de calculs relativement lents pouvant atteindre une heure tandis que l'approche TT ne nécessitent que quelques secondes.
  - **Comparaison entre le champ moyen et la méthode exacte par énumération :** Au vu des scatterplots et des coefficients de corrélation, nous pouvons dire que si nous excluons la structure ordonné où les corrélations sont de -0.499 pour les marginales unaires et -0.218 pour les binaires, la méthode du champ moyen se comporte de manière similaire aux deux autres vis-à-vis de notre référence : les corrélations varient

entre 0.984 et 1 pour les marginales unaires et entre 0.977 et 1 pour les binaires.
L'avantage de cette méthode est qu'elle nécessite des temps de calculs relativement bas par rapport aux deux autres. Cependant, il y a eu plusieurs cas où le schéma du point fixe n'a pas convergé.

- **Comparaison pour un nombre d'individus** $n = 18$: nous prenons désormais comme référence la méthode de Gibbs. Nous obtenons les résultats suivants :
  - **Comparaison entre Gibbs et TT :** Les coefficients de corrélation entre les marginales calculées par Gibbs et l'approche TT varient entre 0.985 et 1 pour les unaires et 0.985 et 1 pour les binaires. De plus, les scatterplot vont vers le sens d'une fonction $X = Y$. Cela indique une très grande proximité entre les deux méthodes.
    En ce qui concerne les temps de calcul, celui de l'approche TT est beaucoup plus raisonnable que celui de Gibbs.
  - **Comparaison entre Gibbs et le champ moyen :** Les résultats indiquent que pour les structures de distances ordonné et dissociative il y a une absence d'accord entre les deux méthodes qui se manifeste dans les faibles corrélations entre les deux méthodes : pour la structure dissociative (resp. ordonné) les corrélations sont -0.500 (resp. 0.693) pour les marginales unaires et -0.125 (resp. 0.688) pour les binaires.
    Pour les quatre autres structures de distances, les corrélations entre les marginales calculées par les deux méthodes varient 0.985 et 1 pour les unaires et entre 0.983 et 1 pour les binaires. Nous pouvons donc en conclure que les méthodes sont en accord pour ces quatre structures.

6.7.2 Cohérence entre les différentes méthodes de calculs des marginales

Nous avons constaté que les mêmes patterns pour les deux deux tailles de problème.

Proximité entre les méthodes

- **Méthode exacte par énumération par rapport aux autres méthodes :** Mise à part le champ moyen dans les structures ordonné et dissociative, les quatre méthodes donnent globalement des résultats très proches.
  Comme nous ne pouvons plus utiliser cette méthode quand le nombre d'individus est grand, cela nous permet de valider la méthode de Gibbs comme notre référence.
- **Approche TT et Gibbs :** Pour les six structures de distances étudiées, nous avons le même constat, que ce soit pour les marginales unaires ou binaires.
  Nous nous penchons maintenant vers les corrélations entre les deux méthodes, nous remarquons des corrélations au-dessus de 0.999 dans les structures associative, Cores periphery et hiérarchique 2. La corrélation minimale est de 0.886 (atteinte entre les marginales binaires calculées dans le cadre de la structure hiérarchique).
  Ces résultats nous permettent de dire que l'approche TT donne de très bons résultats comparés à l'approche de Gibbs qui est réputée très précise en littérature.
- **Approche Point fixe et Gibbs :** L'approche du champ moyen qui est l'approche la plus rapide dans la littérature semble diverger par rapport à la méthode de Gibbs sur deux structures de distances : ordonné et dissociative où nous tombons sur des corrélations très faibles que ce soit pour les marginales binaires ou pour les unaires.
  Cette divergence pour les marginales binaires pourrait s'expliquer par le fait que nous faisons l'hypothèse d'indépendance des classes sachant le graphe ce qui nous conduit à faire le calcul des marginales binaires comme produit des unaires.
  Sur les quatre autres distances, nous observons une très forte corrélation à la fois avec Gibbs.

6.7.3 Temps de calculs/ Précision :

Si nous comparons les temps de calcul, nous réalisons que :

- L'approche de Gibbs nécessite des temps de calculs plus lents étant donné qu'elle exige un nombre important d'itérations pour certaines structures de distances : 12000 pour dissociative et 14000 pour hiérarchique 2.
- L'approche Champ moyen, bien qu'elle soit très rapide, fournit des résultats sont de qualité variable.
- L'approche TT donne des résultats très précis et nécessite des temps de calculs très faibles comparés à Gibbs. Nous pouvons dire que cette méthode donne le meilleur compromis temps de calcul/ précision.

6.7.4 Influence de la structure de distance :

Nos résultats montrent que dans les deux tailles de problème, pour certaines structures de distances toutes les méthodes de calcul approché font des calculs précis des marginales. Il s'agit des structures Cores periphery et hiérarchique.
Pour d'autres structures comme dissociative et ordonnée, les calculs des marginales par le champ moyen est peu précis.
Nous avons aussi constaté que pour la structure dissociative, l'échantillonneur de Gibbs nécessite l'utilisation d'un grand nombre d'échantillons afin d'avoir une bonne précision du calcul des marginales.
Cela suggère que les 6 structures s'ordonnent par précision décroissance avec l'ordre suivant : Cores periphery, hiérarchique, hiérarchique 2, associative, ordonné puis dissociative.
NB : Les résultats que nous avons présentés sont issus de matrices de distances et des facteurs unaires pour lesquels la méthode du point fixe converge.

6.8 Conclusion

6.8.1 Qualité de l'approche TT

Les résultats montrent que l'approche TT est très précise lorsque l'on prend pour référence la méthode exacte par énumération ou l'échantillonneur de Gibbs. Nous avons borné l'erreur des marginales comparé à un calcul exacte dans les formules (B.0.11) et (B.0.11) de l'annexe B qui porte sur la précision des calculs au format TT.
De plus, l'approche TT permet de calculer les marginales dans un temps relativement raisonnable comparé à l'approche de Gibbs.
Nous avons également remarqué que cette approche est stable vis-à-vis de la variation du nombre d'individus et de la structure de distance. Á partir des résultats obtenus, nous pouvons dire que cette approche est celle qui respecte le plus le compromis précision/temps de calcul.

6.8.2 Perspectives

Dans le cadre de ces expériences, nous avons utilisé l'approche TT avec un calcul des marginales comme présentée dans le chapitre 5. Cette approche nécessite beaucoup de stockage et un temps de calcul relativement long. Ceci fait que quand le nombre d'individus augmente, nous ne pourrons plus utiliser cette approche, nous devrons ajouter certaines optimisations que nous allons présenter dans le chapitre 7.

Chapitre 7

Vers le passage à l'échelle de l'approche TT

Table des matières

7.1 Introduction

Dans le chapitre 2, nous avons introduit les SBM qui sont des modèles graphiques très complexes. En effet, comme le graphe des liens entre les sommets est complet, il permet de représenter toutes les interactions entre les individus du graphe, donc deux sommets quelconques sont toujours reliés. Ainsi, si nous avons $n \in \mathbb{N}$ individus, le nombre de liens est $m = \frac{n(n-1)}{2}$. Cela induit une complexité quadratique par rapport au nombre d'individus des calculs d'inférence sur ce type de modèle.
Des difficultés calculatoires apparaissent quand nous appliquons la procédure de Novikov [NROV14] que nous avons expliqué dans le chapitre 5, étant donné que les TT-matrices $\mathbf{A}_i[Z_i]$ et $\mathbf{B}_i$ sont d'ordre m et donc difficilement manipulables quand m est grand. En effet, à chaque somme où produit de ces TT-matrices, nous devrons manipuler m tenseurs (TT-cores) de petites tailles de manière séquentielle.
Cela nous contraint à des optimisations sur cette procédure afin d'éviter de travailler avec des valeurs numériques trop petites, de limiter l'accroissement des rangs des cores ainsi que de réduire le nombre de cores des TT-matrices. Dans la suite de cette section, nous détaillons les trois limitations que nous avons rencontrées lors de l'implémentation de la procédure de Novikov sur des SBM et comment nous y avons répondu.

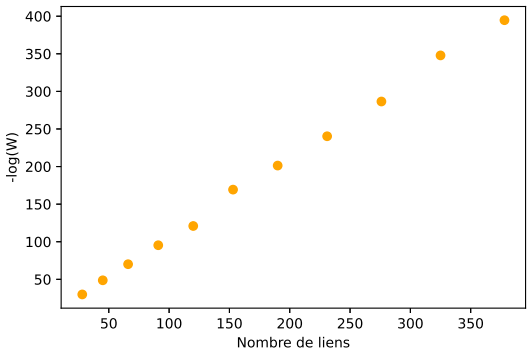
Éviter les problèmes des petites valeurs. Les valeurs des marginales non normalisées et de la constante de normalisation sont très petites. Nous avons constaté que leur logarithme est proportionnel au nombre de liens m de notre modèle, avec le coefficient de proportionnalité négatif.
Dans l'exemple qui suit, nous avons simulé 10 SBM avec des proportions d'individus dans les classes equi-répartis ($\alpha = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$) et des nombres d'individus variant de 8 à 28 par pas de 2. Pour chaque nombre d'individus, nous simulons une matrice de distance et pour chaque modèle, nous calculons la constante de normalisation W.
Nous présentons dans la Figure 7.1 l'évolution de $-\log(W)$ en fonction du nombre de liens $m = \frac{n(n-1)}{2}$ pour deux structures de distances associative et dissociative qui sont des structures classiques pour les SBM et que nous avons présenté dans la section 6.3.

Structure Associative :

$$\Lambda = \begin{pmatrix} 2 & 10 & 10 \\ 10 & 3 & 10 \\ 10 & 10 & 4 \end{pmatrix}$$

Structure Hiérarchique :

$$\Lambda = \begin{pmatrix} 12 & 3 & 1 \\ 3 & 6 & 3 \\ 1 & 3 & 3 \end{pmatrix}$$



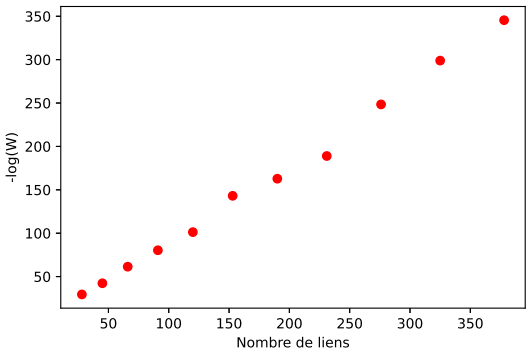


FIGURE 7.1 – Evolution de $-\log(W)$ par rapport au nombre de liens selon une structure de distance associative (à droite) et hiérarchique (à gauche).

Nous calculons la pente des courbes $(m, -\log(W))$ pour chacun des graphes : la pente est de 1.05 pour la structure associative et 0.90 pour la structure hiérarchique. Nous remarquons que l'ordre de grandeur des valeurs de W diffère selon la structure de distances choisie, mais il reste proportionnel à $10^{-m} = 10^{-\frac{n(n-1)}{2}}$.
Á partir de cette remarque, nous pouvons dire que pour des nombres d'individus élevés, les valeurs des constantes de normalisation sont très petites. Cela pose deux problèmes majeurs liés aux limites relatives de la représentation des petites valeurs en arithmétique finie. En effet, quand les valeurs calculées sont très faibles, leur représentation en double précision est difficile et cela conduit à obtenir des valeurs de -0.0 ou 0.0.
En conclusion, à cause des problèmes de représentation des petites valeurs en arithmétique finie, nous nous retrouvons avec des valeurs de la constante de normalisation et des marginales nulles. De plus, lors de la phase LQ de l'algorithme du rounding, nous pouvons obtenir une racine négative ($\sqrt{-0.0}$) ou des logarithmes de valeurs négatives ($\mathrm{Log}\,(-0.0)$) qui engendre soit des NaN (not-a-number) soit des problèmes de numpy RuntimeWarning.
Nous présentons la procédure qui nous permet d'éviter ces problèmes dans la section 7.2.

Explosion des rangs des cores. Dans la méthode proposée par Novikov [NROV14], le calcul de la constante de normalisation, qui est très similaire à celui des marginales (détaillé dans la section 5.2.6) est décrit dans l'algorithme 1.

Algorithm 1 Calcul de W

Require: n nombre d'individus, $[\mathbf{B}_1, \ldots, \mathbf{B}_n]$: liste de TT-matrices intervenant dans le calcul, ϵ : précision du rounding.
Ensure: $\mathbf{W}$: constante de normalisation.

1: $\mathbf{W} = \mathbf{B}_n$
2: **for** $i = n-1, \ldots, 1$ **do**
3: $\quad \mathbf{W} = \mathbf{B}_i \times \mathbf{W}$
4: $\quad \mathbf{W} = \mathrm{rounding}(\mathbf{W}, \epsilon)$
5: **end for**
6: Transformation de $\mathbf{W}$ en scalaire W
7: **return** W

Dans l'algorithme 1, nous commençons par attribuer le TT-vecteur $\mathbf{B}_n$ à $\mathbf{W}$. Ensuite, chaque itération se décompose en deux phases, une phase de produits de TT-matrices-TT-vecteur (ligne 3 de l'algorithme 1) où nous calculons le produit $\mathbf{B}_n \ldots \mathbf{B}_{i+1} \times \mathbf{B}_i$ et une phase de compression (ligne 4 de l'algorithme 1) où nous faisons la compression du TT-vecteur résultant par l'algorithme du rounding.
Nous faisons ces deux phases étant donné que chaque produit entre deux TT-matrices augmente le rang de la TT-matrice résultante et que si nous faisons les calculs sans recompression du résultat (ligne 4) à chaque itération, les rangs des TT-cores qui composent les TT-matrices augmenteront jusqu'à atteindre la valeur théorique de Q^n alors que la constante de normalisation est un scalaire.
Afin d'éviter ces rangs élevés, nous sommes contraints de recompresser le TT-vecteur résultant après chaque itération comme nous présentons dans la ligne 4 de l'algorithme 1 afin de limiter la croissance des rangs des TT-cores.

Dans la procédure de Novikov et al., la compression est faite avec l'algorithme du rounding avec une précision ϵ fixée, cela nous permet en plus de contrôler la précision du résultat final. Cependant, faire la compression de cette manière n'empêche pas les rangs de croître fortement quand n augmente.

Pour illustrer nos propos, nous calculons la constante de normalisation par l'intermédiaire de l'algorithme 1 pour un modèle SBM avec comme paramètres : $n = 22$, $Q = 3$, $\Lambda = \begin{pmatrix} 2 & 10 & 10 \\ 10 & 3 & 10 \\ 10 & 10 & 4 \end{pmatrix}$ et $\alpha = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$.

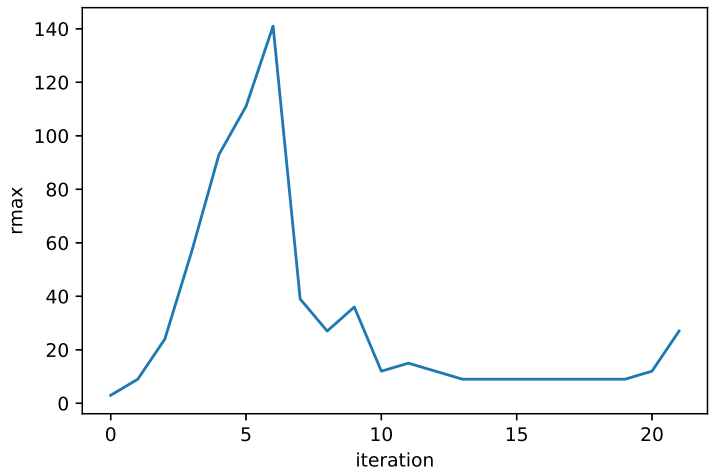


FIGURE 7.2 – Évolution de rang dans le calcul de $\mathbf{W}$ en fonction de l'itération.

La Figure 7.2 présente l'évolution du rang maximal des cores de W en fonction des itérations de l'algorithme 1 avec le paramètre de compression ϵ fixé à 5×10^{-2}. Nous observons que les rangs augmentent jusqu'à atteindre 140 avant de diminuer pour atteindre 35 lors de la dernière itération. Sans rounding ce calcul n'est pas possible car le TT-rang maximal est : $3^{22} = 31\,381\,059\,609$. Nous présentons la procédure qui nous permet de réduire les rangs des cores des TT-matrices dans la section 7.3.

Réduction du nombre de cores. Pour un modèle SBM avec n individus, nous avons m liens. Chacune des TT-matrices (ou TT-vecteurs) $\mathbf{A}_i[Z_i]$ et $\mathbf{B}_i$ est d'ordre m. Cela a pour conséquences : Chaque opération (somme ou produit) sur les TT-matrices $\mathbf{A}_i[Z_i]$ et $\mathbf{B}_i$ nécessite d'itérer séquentiellement sur les m cores (détaillé dans la section 4.4.2).
De plus, nous avons expliqué dans la section 4.4.4 que pour une TT-matrice composée de m cores, l'algorithme du rounding consiste à faire une orthogonalisation (décomposition QR) puis une décomposition en valeur singulière (SVD) pour chaque core, le tout en parcourant séquentiellement toutes les dimensions.
Ces deux raisons font que nous avons besoin de trouver une manière de diminuer l'ordre du tenseur afin de réduire la taille de la boucle sur les dimensions ainsi que le temps du calcul des produits des TT-matrices (ligne 3 de l'algorithme 1) et de la compression par rounding (ligne 4 de l'algorithme 1) à chaque itération.
Nous présentons la procédure qui nous permet de diminuer l'ordre du tenseur dans la section 7.4.

7.2 Procédure d'homothétie

Nous avons présenté auparavant l'évolution des valeurs obtenues pour la constante de normalisation et la figure 7.1 indique qu'elles sont proportionnelles à 10^{-m}.

Cela ne gêne pas d'un point de vue mathématique, par contre, pour l'implémentation des calculs, les nombres sont stockés soit en simple, soit en double précision. Dans la section qui suit, nous expliquons comment se fait la représentation informatique des valeurs en double précision. Nous présentons ensuite une solution basée sur une homothétie pour s'affranchir de ce problème.

7.2.1 Représentation en double précision

En informatique, les nombres sont représentés selon le standard IEEE 754 [46108], celle-ci stipule qu'un réel v est représenté en double précision (resp. simple precision) par sa mantisse, son exposant et son signe en 64 bits (resp. 32 bits) selon la figure a (resp. b) du schéma 7.3 (figure 5 de l'article [NLTK19]) :

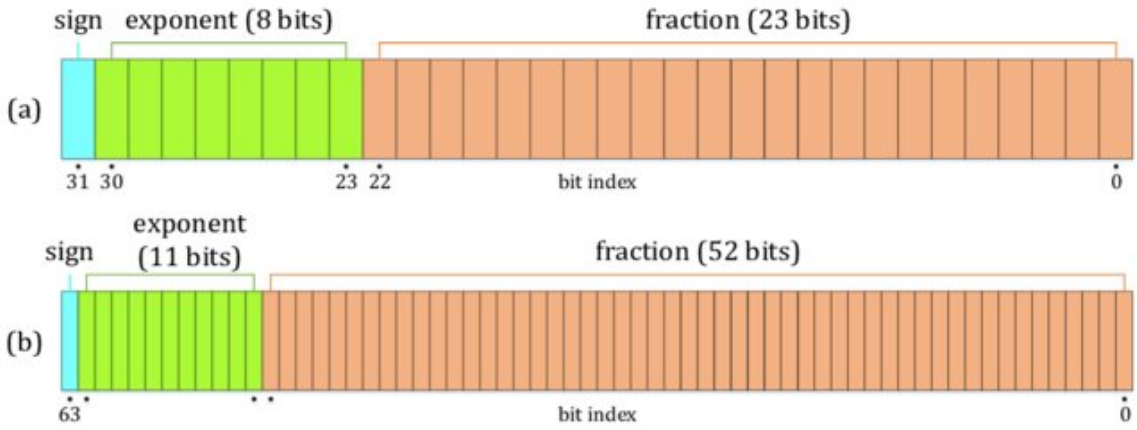


FIGURE 7.3 – Standard IEEE 754 pour la double précision.

Ce format de stockage en 64 bits des nombres se décompose en trois parties : 1 bit pour le stockage du signe, 52 bits pour le stockage de la mantisse et 11 bits pour le stockage de l'exposant.

Le réel v est représenté en nombre normalisé de cette manière :

$$v = s \times (1 + ma) \times 2^{(e-d)} \tag{7.2.1}$$

Avec,

- $s \in \{-1, +1\}$ représente le signe de v
- e exposant.
- d décalage.
- ma représente la mantisse $0 \leq ma \leq 1$.

Á partir de la formule (7.2.1), le plus petit nombre positif normalisé qui peut être représenté est : $2^{-(2^{11-1}-2)} = 2^{-(2^{10}-2)} \approx 2.22 \times 10^{-308}$.
Il existe une autre représentation des réels appelée représentation en nombres dénormalisés, une manière de représenter des nombres ayant une valeur très proche de zéro. Dans cette représentation, les nombres vérifient les conditions suivantes : $ma \neq 0$ et $d = 0$.
L'exposant devient baisé par $2^{11-1} - 1$ alors la formule (7.2.1) pour stocker les nombres devient :

$$v = s \times (1 + ma) \times 2^{(-d)} \tag{7.2.2}$$

Á partir de la formule (7.2.2), la plus petite valeur qu'on peut représenter est :

$$v_{min} = 2^{-(2^{se-1}-2+sm)} = 2^{-1074} \approx 5 \times 10^{-324} \tag{7.2.3}$$

Remarque : Au niveau de l'implémention nous utilisation la représentation en nombre dénormalisé [SST03] vue qu'elle nous offre plus de flexibilité et donc le réel le plus petit que nous pouvons représenter est 5×10^{-324}.

7.2.2 Motivations

Comme nous travaillons avec des petites valeurs, cela présente deux difficultés majeures pour l'implémentation des calculs présentés dans la section 5.

- Premièrement, comme les valeurs des marginales et des constantes de normalisation sont très petites, alors il est difficile de les représenter en arithmétique finie et donc une valeur de 0.0 ou -0.0 leur est attribuée.
- Deuxièmement, ses valeurs très basses induisent des erreurs au niveau de la phase d'orthogonalisation lors de la compression via l'algorithme du rounding. Lors de la phase d'orthogonalisation, nous sommes confrontés à des cores dont les normes sont tellement petites qu'ils sont considérés comme nul et d'autres avec une norme égale à 1.
  Pour illustrer ces problèmes, nous partons d'une TT-matrice $\mathbf{A}$ qui s'exprime :

  $$\mathbf{A}[i_1, \ldots, i_d\,; j_1, \ldots, j_d] = \mathbf{G}_1(i_1, j_1).\mathbf{G}_2(i_2, j_2).\ldots.\mathbf{G}_d(i_d, j_d)$$

  avec, $k \in [\![1, d]\!]$, $\quad \mathbf{G}_k$ le k-ième core de $\mathbf{A}$.
  Si les normes de ces cores sont très faibles, c'est-à-dire proches de v_{min}, alors, pour la phase d'orthogonalisation du rounding, les décompositions LQ des cores $\mathbf{G}_k$ donnent des matrices Q de norme 1 et des matrices L de valeurs très faibles.
  Á cause de cela, nous nous retrouvons à faire des opérations comme les sommes, les soustractions, les division et les racines sur des valeurs très faibles. Cela conduit à des problèmes d'instabilités numériques, des divisions par zéros et des NaN.

Afin de remédier à ces problèmes, nous allons faire une homothétie sur les facteurs.

7.2.3 Procédure mise en oeuvre

La procédure d'homothétie consiste à multiplier par un scalaire chaque facteur $\psi'_{i,j}$ défini par l'équation (5.2.5).
Soit $a \in \mathbb{R}$, nous définissons les facteurs normalisés par $\psi^p_{i,j}$ comme suit

$$\forall i \in [\![1, n-1]\!], j \in [\![i+1, n]\!], \qquad \psi^p_{i,j}(z_i, z_j) = a\psi'_{i,j}(z_i, z_j) \tag{7.2.4}$$

Proposition 8 (Homothétie des facteurs)**.** *Si nous multiplions tous les facteurs par une constante a et que nous mettons ces facteurs à l'entrée de la procédure de Novikov, nous obtenons la nouvelle constante de normalisation $\overline{W}$ agrandie par rapport à W de cette manière :*

$$\overline{W} = a^m W$$

et les nouvelles marginales non normalisées

$$\overline{p}_{i,j}(z_i, z_j) = a^m p_{i,j}(z_i, z_j)$$

Démonstration. On définit la loi non normalisée $\mathbb{P}_\theta(z, D)$ comme le produit de facteurs

$$\mathbb{P}(z, D) = \left[\prod_{i=1}^{n} \prod_{j>i} \psi'_{i,j}(z_i, z_j)\right]$$

et la constante de normalisation s'écrit

$$W = \sum_z \mathbb{P}(z, D)$$

Nous pouvons exprimer $\mathbb{P}(z, D)$ de cette manière ;

$$\mathbb{P}(z, D) = \left[\prod_{i=1}^{n} \prod_{j>i} \frac{\psi_{i,j}^{p}(z_i, z_j)}{a} \right]$$

Alors,

$$\mathbb{P}(z, D) = \left[\prod_{i=1}^{n} \prod_{j>i} \psi_{i,j}^{p}(z_i, z_j) \right] \prod_{i=1}^{n} \prod_{j>i} \left[\frac{1}{a} \right]$$

$$= \left[\prod_{i=1}^{n} \prod_{j>i} \psi_{i,j}^{p}(z_i, z_j) \right] a^{\sum_{i=1}^{n} \sum_{i>j} -1}$$

$$\mathbb{P}(z, D) = \left[\prod_{i=1}^{n} \prod_{j>i} \psi_{i,j}^{p}(z_i, z_j) \right] a^{-m}$$

Si nous posons :

$$\overline{\mathbb{P}}(z, D) = \left[\prod_{i=1}^{n} \prod_{j>i} \psi_{i,j}^{p}(z_i, z_j) \right]$$

Alors,

$$\mathbb{P}(z, D) = a^{-m} \overline{\mathbb{P}}(z, D)$$

Ensuite,

$$W = \sum_z \mathbb{P}(z, D) = a^{-m} \sum_z \overline{\mathbb{P}}(z, D) = a^{-m} \overline{W}$$

Nous aurons la même chose pour les marginales binaires et unaires non normalisées :

$$p_{i,j}(z_i, z_j) = \sum_{z \setminus z_i, z_j} \mathbb{P}(z, D) = a^{-m} \sum_{z \setminus z_i, z_j} \overline{\mathbb{P}}(z, D) = a^{-m} \overline{p}_{i,j}(z_i, z_j)$$

□

Si nous normalisons les marginales par W, nous perdons la constante d'homothétie. En effet :

$$\frac{\overline{p}_{i,j}(z_i, z_j)}{\overline{W}} = \frac{a^m p_{i,j}(z_i, z_j)}{a^m W} = \frac{p_{i,j}(z_i, z_j)}{W} \tag{7.2.5}$$

Pour résumer cette procédure, nous augmentons numériquement les facteurs afin d'éviter les différents problèmes de petites valeurs puis nous retrouvons les vraies valeurs des marginales après normalisation.
Si nous nous basons sur les figures 7.1 présentées durant l'introduction, la pente des courbes nous suggère qu'il faut utiliser une constante d'homothétie $a = 10$.
Cette constante n'est pas la meilleure, mais elle nous permet d'éviter des valeurs très basses pour la constante de normalisation (inférieures à 10^{-324}).

7.2.4 Conclusion

La valeur de la constante de normalisation peut facilement atteindre des ordres de grandeurs inférieurs à 10^{-325} au cas où le nombre d'individus dépasse 26, cela conduit à des problèmes de représentation des petits chiffres, ce qui cause des problèmes de stabilités numériques ainsi que des problèmes de NaN (Not a Number) lors de la phase d'orthogonalisation dans le rounding.
L'approche par homothétie nous a permis d'atteindre le cap de $n = 26$ pour les calculs des marginales et $n = 33$ pour la constante de normalisation.
Cependant, nous ne pouvons pas utiliser le graphe présenter dans la section 5.3 à cause de l'explosion des rangs des cores des TT-matrices.
Dans ce qui va suivre, nous présentons une procédure qui nous permet de dépasser cette limite.

7.3 Changement de paramètres dans la compression par l'algorithme du rounding

Dans la proposition 2, nous avons montré que le produit de TT-matrices (resp. une TT-matrice par un TT-vecteur) augmente le rang de la TT-matrice (resp. le TT-vecteur) résultante. Pour un modèle SBM avec n individus, en utilisant l'algorithme 1, le calcul de la constante de normalisation ou d'une marginale nécessite de faire $n-1$ produits TT-matrices TT-vecteur.
Nous avons indiqué durant l'introduction que l'algorithme classique du rounding ne suffit pas à limiter la croissance des rangs des TT-vecteurs obtenus après chaque itération de l'algorithme 1. Cela a pour effet d'augmenter l'espace nécessaire de stockage du tenseur ainsi que le temps de calcul. Nous montrons dans la section 7.3.2 que la méthode classique du rounding présentée durant la section 4.4.4 est insuffisante quand le nombre d'individus augmente et ne nous permet pas d'avoir un temps de calcul raisonnable.
Puis dans la section 7.3.3, nous regardons comment nous pouvons changer les paramètres de l'algorithme du rounding afin d'accélérer les calculs et nous donnons des résultats numérique dans la section 7.3.4.

7.3.1 Choix de la précision du rounding

Afin de définir la précision du rounding que nous utilisons pour le reste du chapitre, nous avons pris cinq valeurs $[1\times10^{-1}, 1\times10^{-2}, 5\times10^{-2}, 1\times10^{-3}, 1\times10^{-4}]$.
Pour chacune de ces valeurs, nous définissons une structure de distance Λ pour chacune des six structures définies dans la section 6.3. Ensuite, nous avons simulé des SBM avec des nombres d'individus n variant entre 8 et 20 par pas de deux. Pour chacun de ces modèles, nous avons défini le nombre de classes $Q = 3$ et les proportions d'individus dans les classes $\alpha = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$.
Pour chaque nombre d'individus, nous simulons la matrice de distances D et nous appliquons la procédure de Novikov, nous calculons les TT-matrices $\mathbf{B}_i$. Enfin, nous utilisons l'algorithme 1 avec une précision parmi les cinq valeurs et nous considérons comme référence $\epsilon = 1\times10^{-4}$
Pour la structure associative, le tableau 7.1 présente les constantes de normalisation et le tableau 7.2 présente les temps de calculs.

Nous observons que toutes les valeurs de la constante de normalisation obtenues sont proches de celle obtenue par le ϵ de référence à part celle obtenue pour un modèle SBM avec $n = 12$ et pour une précision $\epsilon = 1\times10^{-1}$.

n \ ϵ	1×10^{-1}	5×10^{-2}	1×10^{-2}	1×10^{-3}	1×10^{-4}
8	1.30×10^{-30}	1.30×10^{-30}	1.30×10^{-30}	1.30×10^{-30}	1.30×10^{-30}
10	1.18×10^{-52}	1.2×10^{-52}	1.2×10^{-52}	1.2×10^{-52}	1.2×10^{-52}
12	$\mathbf{9.54\times10^{-74}}$	1.09×10^{-73}	1.09×10^{-73}	1.09×10^{-73}	1.09×10^{-73}
14	1.37×10^{-102}	1.37×10^{-102}	1.37×10^{-102}	1.37×10^{-102}	1.37×10^{-102}
16	9.73×10^{-138}	9.72×10^{-138}	9.80×10^{-138}	9.8×10^{-138}	9.8×10^{-138}
18	4.32×10^{-163}	4.32×10^{-163}	4.35×10^{-163}	4.35×10^{-163}	4.35×10^{-163}
20	1.94×10^{-203}	1.96×10^{-203}	1.96×10^{-203}	1.96×10^{-203}	1.96×10^{-203}
22	4.92×10^{-255}	4.92×10^{-255}	4.93×10^{-255}	4.93×10^{-255}	4.93×10^{-255}

TABLE 7.1 – Constante de normalisation pour différentes valeurs de ϵ dans le cas d'une structure associative

	1×10^{-1}	5×10^{-2}	1×10^{-2}	1×10^{-3}	1×10^{-4}
8	0.010712	0.015826	0.023264	0.054599	0.134134
10	0.113050	0.173764	0.310835	1.203731	3.219305
12	0.202726	0.339975	2.218524	5.057278	12.831984
14	0.249002	0.329584	0.759825	2.940815	8.567263
16	0.546148	0.790832	1.437784	3.304456	8.719653
18	0.241273	0.290420	1.043211	5.462873	19.105516
20	0.220903	0.271335	0.430281	1.848418	5.202058
22	1.042942	1.579870	14.846940	15.378039	63.950157

TABLE 7.2 – Temps de calcul de la constante de normalisation pour différentes valeurs de ϵ dans le cas d'une structure associative

Nous montrons dans les résultats que le rounding avec $\epsilon = 5\times10^{-2}$ est largement suffisant pour avoir une bonne précision de la constante de normalisation. De plus, il nous permet d'avoir un très bon temps de calcul comparé aux autres paramètres de rounding.

7.3.2 Explication du problème

Dans cet exemple, nous utilisons un modèle SBM avec comme paramètres : $n = 7$, $Q = 3$ $\Lambda = \begin{pmatrix} 2 & 10 & 10 \\ 10 & 3 & 10 \\ 10 & 10 & 4 \end{pmatrix}$ et $\alpha = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. Nous simulons les classes des individus et la matrice de distances D comme détaillé dans la section 6.4.1. Ensuite, nous calculons la constante de normalisation W sans faire la compression par rounding puis avec compression comme présenté dans l'algorithme 1. Pour chaque itération des calculs, nous traçons sur les figures 7.4 et 7.5 les rangs des cores.
Pour les tracés de rangs, à l'itération 1, nous traçons les rangs des 21 cores de $\mathbf{B}_n$, pour l'itération $i-1$ nous traçons les rangs des 21 cores de $\mathbf{B}_n \dots \mathbf{B}_{n-i}$.
Dans la figure 7.4, nous traçons les rangs des cores des TT-matrices calculées à chaque itération de l'algorithme 1 sans la recompression (ligne 4 de l'algorithme). Nous observons que les rangs de la plupart des cores évoluent de manière exponentielle : ils valent Q^i pour l'itération i et atteignent une valeur de Q^n pour la dernière itération. Dans cet exemple, il s'agit de $3^7 = 2187$.

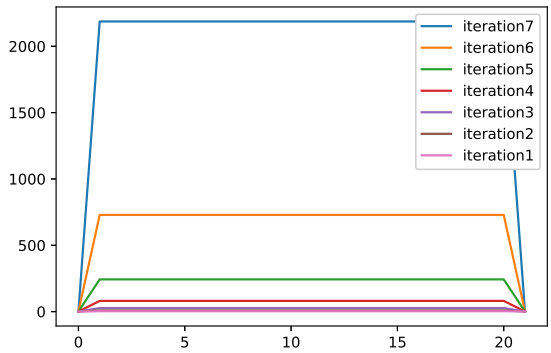


FIGURE 7.4 – Évolution des rangs des cores dans un calcul sans rounding pour $n = 7$.

Nous réitérons l'expérience pour les mêmes paramètres et matrice de distance en recompressant le tenseur W à chaque itération (ligne 4 de l'algorithme 1) avec une précision de $\epsilon = 5 \times 10^{-2}$. La figure 7.5 indique l'évolution du rang de chacun des cores en fonction de l'itération et le tableau 7.3 présente les rangs des cores.

It \ core	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	1
2	1	1	1	1	1	3	9	9	9	9	9	9	9	9	9	9	9	9	9	3	1
3	1	1	1	1	3	8	19	19	19	19	19	19	19	19	19	19	9	3	1	1	1
4	1	1	1	3	9	19	30	30	30	31	32	**33**	17	9	3	1	1	1	1	1	1
5	1	1	3	3	7	10	12	12	9	5	3	1	1	1	1	1	1	1	1	1	1
6	1	3	6	6	5	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

TABLE 7.3 – Tableau présentant les rangs des cores des TT-matrices obtenues à chaque itération

Dans le tableau 7.3, chaque ligne correspond à l'itération de l'algorithme 1 et chaque colonne au numéro du core. Nous observons que lors de la première itération, les cores sont de rang 1 ou 3. Puis lors des premières itérations, ils augmentent avant de décroitre à partir de la cinquième itération jusqu'à valoir 1 lors de la dernière.

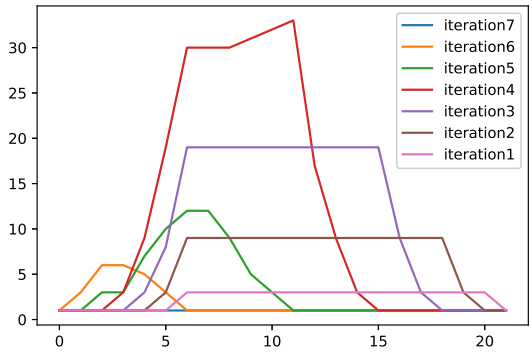


FIGURE 7.5 – Évolution des rangs des cores dans un calcul avec rounding à $\epsilon = 0.05$ pour $n = 7$.

Dance le graphique 7.5, nous constatons qu'après chaque itération de l'algorithme 1 le rang

augmente jusqu'à la quatrième itération jusqu'à atteindre 33, soit à peine plus que $Q^3 = 27$. Ensuite, les rangs des cores baissent jusqu'à atteindre 1 lors de la dernière itération. De plus, nous remarquons qu'après chaque itération, le nombre de cores de rangs 1 augmente et qu'à la dernière itération tous les cores sont de rang 1.

Nous comparons désormais les précisions et les temps de calculs deux expériences présentées dans les graphes 7.4 et 7.5. Nous observons :

- **Erreur relative.** Nous comparons les résultats obtenus pour chaque calcul et nous obtenons une erreur relative de 1.31×10^{-7} alors que nous souhaitions 5×10^{-2}.
- **Temps de calcul.** Le temps de calcul pour la première expérience est de 1.09 secondes, tandis qu'il est de 0.05 secondes pour la deuxième expérience.

Cette expérience justifie l'importance de faire des compressions par rounding à chaque itération de les calculs. De plus, quand le nombre d'individus est égal à 8, le calcul de la constante de normalisation devient difficile étant donné que nous devons manipuler des TT-matrices avec des cores de rangs $3^8 = 6561$.

Nous refaisons la même expérience présentée dans le graphe 7.5, mais nous augmentons le nombre d'individus n à 22. Nous aurons un nombre de liens $m = 231$. Nous traçons ensuite l'évolution du rang des cores en fonction des itérations dans la figure 7.6.

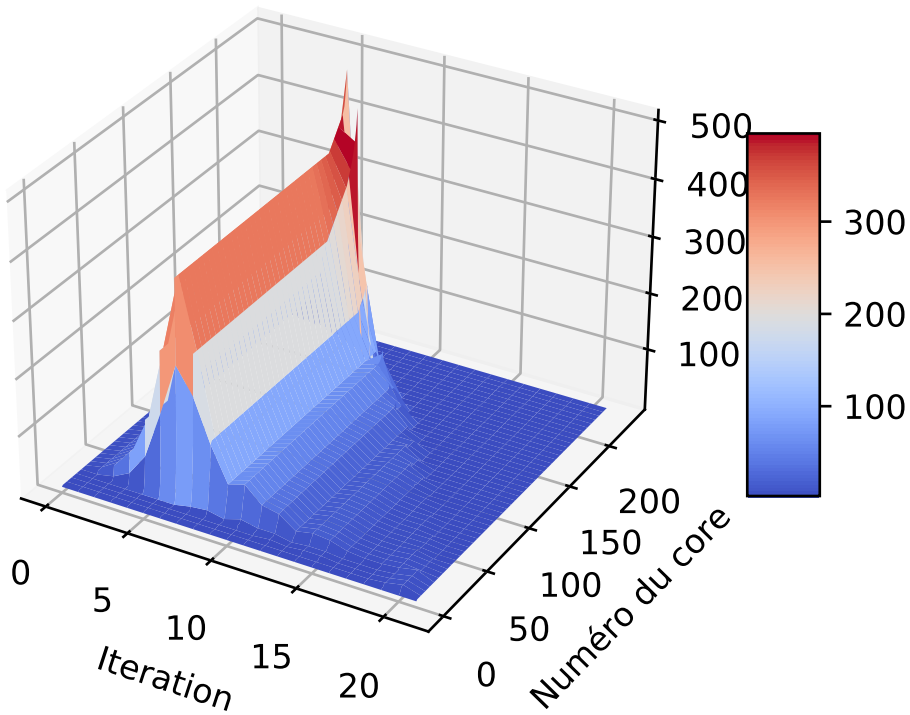


FIGURE 7.6 – Évolution des rangs des cores dans un calcul avec compression avec l'algorithme du rounding à $\epsilon = 5 \times 10^{-2}$ pour un modèle SBM avec $n = 22$.

Dans la figure 7.6, nous avons tracé une surface 3D, une représentation tridimensionnelle des rangs des cores (l'axe des z) en fonction de son ordre (l'axe des y) et des itérations de l'algorithme 1 (l'axe des x). Nous faisons varier les couleurs du bleu (pour les valeurs basses des rangs) au rouge (pour les valeurs hautes des rangs).
Nous constatons que durant les premières itérations, les rangs des cores de la TT-matrice résultante croient jusqu'à atteindre 507 lors de la septième itération avant de décroître au fur et à mesure des itérations jusqu'à atteindre 1 pour la plupart des cores lors de la dernière itération.
Á partir de cet exemple, nous pouvons dire que si le nombre d'individus augmente, le calcul de la constante de normalisation devient quasiment impossible vu que le stockage nécessaire est trop grand.

Nous avons illustré dans les figures 7.4 et 7.5 l'intérêt de faire une compression par le rounding. Cependant, dès que nous augmentons le nombre d'individus, les rangs explosent, même avec rounding avec comme paramètre ϵ. Et donc, le calcul de la constante de normalisation à cette précision devient très lent voir impossible dès que le nombre d'individus est supérieur à 33.
Dans les sous-sections suivantes, nous présentons comment changer les paramètres du rounding afin d'accélérer les calculs sans pour autant perdre en termes de précision.

7.3.3 Présentation du rounding

Dans le cadre de l'approche TT, nous n'utilisons que des TT-vecteurs obtenus par produits de TT-matrice TT-vecteurs comme nous l'avons présenté dans la sous-section 5.2.6.
Á chaque itération, nous appliquons le rounding que nous avons expliqué dans la sous-section 4.4.4 et que nous rappelons dans cette sous-section.
Pour une TT-matrice donnée, nous pouvons réduire les rangs de leurs cores avec deux méthodes de recompression par l'algorithme du rounding présentées dans la section 4.4.4.

- **TT-rounding avec précision prescrite** ϵ où nous pouvons contrôler la précision de l'approximation de la TT-matrice, mais pas le rang de ses cores.
- **TT-rounding avec un rang prescrit** où nous pouvons contrôler les rangs des TT-cores de l'approximation de la TT-matrice, mais pas sa précision.

Dans les différentes librairies python que nous avons utilisé dans le cadre de la manipulation des opérations sur les TT-matrices et TT-vecteur, il existe une méthode alternative aux deux premières. Cette méthode prend comme paramètres une précision ϵ et un rang prescrits et fait une compression des TT-matrices ou des TT-vecteurs en contrôlant le rang tout en gardant une meilleure précision que le TT-rounding avec un rang prescrit.
Nous appliquons cette méthode pour la compression des TT-matrices et nous la comparons avec les deux autres.

7.3.4 Résultats

7.3.4.1 Calcul de la constante de normalisation

Pour l'ensemble des expériences que nous présentons, nous avons fixé les paramètres suivants : Le nombre de classes $Q = 3$ et les proportions d'individus dans les classes $\alpha = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$. Nous faisons varier la structure de la matrice des distances Λ parmi les six présentées dans la section 6.4.

Pour chaque structure, nous faisons varier le nombre d'individus n entre 8 et 32 par pas de deux. Pour chaque nombre d'individus, nous simulons la matrice de distances D et nous appliquons

la procédure de Novikov avec notre procédure d'homothétie pour calculer les TT-matrices $\mathbf{B}_i$. Enfin, nous fixons $\epsilon = 5 \times 10^{-2}$ et nous calculons la constante de normalisation par six méthodes. Premièrement avec ϵ fixé, puis $r_{max} = 9$ et ϵ fixé, $r_{max} = 3$ et ϵ fixé, $r_{max} = 9$, $r_{max} = 27$ et ϵ fixé et enfin $r_{max} = 81$ et ϵ fixé. Les résultats présentés sont les suivants :

- Premièrement, nous présentons dans les tableaux 7.5 et 7.7 les résultats du calcul de la constante de normalisation.
  Nous présentons aussi le plus grand TT-rang max observé durant les différentes itérations de l'algorithme 1 lorsque le rounding prend comme paramètre uniquement la précision ϵ.
- Deuxièmement, nous présentons les temps de calcul pour les différents calculs dans les tableaux 7.6 et 7.8 .

NB : Nous ne présentons que les structures associative et cores periphery dans ce chapitre, les quatre autres structures sont présentés dans l'annexe A.
Pour une meilleure lisibilité des tableaux, nous mettrons en gras les résultats qui diffèrent de la valeur de référence qui est celle obtenue lorsqu'on fait un calcul avec rounding avec $\epsilon = 5 \times 10^{-2}$ fixé. Pour les discussions et les commentaires des résultats, nous notons les paramètres du rounding de cette manière :

	ϵ	$r_{max} = 3$ et ϵ	$r_{max} = 9$ et ϵ	$r_{max} = 9$	$r_{max} = 27$ et ϵ	$r_{max} = 81$ et ϵ
	R_1	R_2	R_3	R_4	R_5	R_6

TABLE 7.4 – Tableau des abréviations pour différentes méthodes de rounding.

Associative

	R_1 ($r_{\max}$)	R_2	R_3	R_4	R_5	R_6
n=8	6.21×10^{-28} (9)	2.68×10^{-28}	6.21×10^{-28}	6.26×10^{-28}	6.21×10^{-28}	6.21×10^{-28}
n=10	3.2×10^{-45}(12)	2.24×10^{-46}	3.20×10^{-45}	3.20×10^{-45}	3.2×10^{-45}	3.2×10^{-45}
n=12	1.0×10^{-75}(31)	$\mathbf{3.32 \times 10^{-77}}$	$\mathbf{6.73 \times 10^{-76}}$	$\mathbf{6.78 \times 10^{-76}}$	1.0×10^{-75}	1.0×10^{-75}
n=14	5.38×10^{-101}(8)	$\mathbf{1.11 \times 10^{-101}}$	5.38×10^{-101}	5.39×10^{-101}	5.38×10^{-101}	5.38×10^{-101}
n=16	3.30×10^{-135}(10)	3.05×10^{-135}	3.30×10^{-135}	3.34×10^{-135}	3.30×10^{-135}	3.30×10^{-135}
n=18	2.05×10^{-161}(15)	$\mathbf{1.14 \times 10^{-161}}$	2.05×10^{-161}	2.05×10^{-161}	2.05×10^{-161}	2.05×10^{-161}
n=20	1.18×10^{-199}(18)	$\mathbf{2.65 \times 10^{-205}}$	1.18×10^{-199}	1.18×10^{-199}	1.18×10^{-199}	1.18×10^{-199}
n=22	1.59×10^{-247}(15)	1.80×10^{-247}	1.59×10^{-247}	1.59×10^{-247}	1.59×10^{-247}	1.59×10^{-247}
n=24	9.57×10^{-294}(16)	$\mathbf{1.12 \times 10^{-297}}$	9.57×10^{-294}	9.57×10^{-294}	9.57×10^{-294}	9.57×10^{-294}
n=26	1.80×10^{-353} (16)	$\mathbf{8.13 \times 10^{-375}}$	1.80×10^{-353}	1.80×10^{-353}	1.80×10^{-353}	1.80×10^{-353}
n=28	1.34×10^{-393} (21)	1.09×10^{-393}	1.34×10^{-393}	1.34×10^{-393}	1.34×10^{-393}	1.34×10^{-393}
n=30	4.03×10^{-459} (21)	4.03×10^{-459}	4.03×10^{-459}	4.03×10^{-459}	4.03×10^{-459}	4.03×10^{-459}
n=32	3.43×10^{-511} (34)	3.19×10^{-511}	3.43×10^{-511}	3.43×10^{-511}	3.43×10^{-511}	3.43×10^{-511}

TABLE 7.5 – Constante de normalisation calculée ainsi que le $r_{\max}$ observé pour la méthode R_1 dans le cas associative.

En termes de résultats obtenus et de temps de calculs, la structure associative est la structure la plus simple parmi les six structures utilisées pour simuler les matrices de distances. Nous constatons que seul le rounding R_2 donne des résultats éloignés de notre référence (R_1).
En termes de temps de calcul, le rounding R_3 est beaucoup plus rapide si nous excluons R_2. Par contre, nous remarquons qu'il fait une erreur dans le calcul des de la constante de normalisation pour $n = 12$.

	R_1	R_2	R_3	R_4	R_5	R_6
n=8	0.05	0.00	0.01	0.01	0.02	0.02
n=10	0.09	0.01	0.04	0.04	0.04	0.04
n=12	0.28	0.01	0.05	0.06	0.14	0.14
n=14	0.33	0.01	0.07	0.10	0.16	0.16
n=16	1.02	0.02	0.06	0.15	0.48	0.50
n=18	0.52	0.02	0.10	0.22	0.26	0.26
n=20	0.33	0.04	0.14	0.32	0.17	0.17
n=22	2.08	0.04	0.17	0.42	0.81	0.93
n=24	0.41	0.06	0.15	0.55	0.19	0.19
n=26	1.79	0.07	0.38	0.72	0.85	0.83
n=28	0.91	0.07	0.32	0.94	0.42	0.42
n=30	5.52	0.08	0.37	1.16	2.35	2.36
n=32	2.65	0.10	0.32	1.38	1.17	1.17

TABLE 7.6 – Temps de calcul en secondes pour W dans le cas associative.

Au vu de ces résultats, R_5 conduit au meilleur rounding pour obtenir des résultats aussi précis que notre référence bien qu'il soit plus coûteux que les autres.
Nous pouvons aussi remarquer que les temps de calcul entre R_5 et R_6 sont similaires, cela est lié aux valeurs assez faibles du $r_{\max}$ comprises entre 9 et 34, cela fait que faire ces rounding reviennent à appliquer R_1.

Cores periphery

	R_1 ($r_{\max}$)	R_2	R_3	R_4	R_5	R_6
n=8	2.63×10^{-33} (58)	$\mathbf{7.16 \times 10^{-36}}$	$\mathbf{6.90 \times 10^{-34}}$	$\mathbf{7.33 \times 10^{-34}}$	$\mathbf{1.55 \times 10^{-33}}$	2.62×10^{-33}
n=10	1.55×10^{-46} (99)	$\mathbf{3.70 \times 10^{-47}}$	$\mathbf{9.87 \times 10^{-47}}$	$\mathbf{1.01 \times 10^{-46}}$	1.33×10^{-46}	1.54×10^{-46}
n=12	2.12×10^{-70} (34)	1.67×10^{-70}	2.01×10^{-70}	2.01×10^{-70}	2.12×10^{-70}	2.12×10^{-70}
n=14	1.60×10^{-92} (30)	$\mathbf{5.68 \times 10^{-102}}$	1.57×10^{-92}	1.57×10^{-92}	1.60×10^{-92}	1.60×10^{-92}
n=16	5.83×10^{-129} (43)	$\mathbf{1.36 \times 10^{-137}}$	$\mathbf{1.38 \times 10^{-137}}$	$\mathbf{2.26 \times 10^{-137}}$	5.83×10^{-129}	5.83×10^{-129}
n=18	2.47×10^{-154} (38)	$\mathbf{5.63 \times 10^{-171}}$	2.47×10^{-154}	2.47×10^{-154}	2.47×10^{-154}	2.47×10^{-154}
n=20	6.88×10^{-188} (89)	$\mathbf{1.27 \times 10^{-190}}$	5.66×10^{-188}	5.55×10^{-188}	6.53×10^{-188}	6.62×10^{-188}
n=22	4.06×10^{-238} (83)	$\mathbf{1.03 \times 10^{-253}}$	$\mathbf{1.71 \times 10^{-252}}$	$\mathbf{8.75 \times 10^{-250}}$	3.71×10^{-238}	4.06×10^{-238}
n=24	2.41×10^{-280}(135)	$\mathbf{4.80 \times 10^{-308}}$	2.41×10^{-280}	2.46×10^{-280}	2.41×10^{-280}	2.41×10^{-280}
n=26	5.17×10^{-324}(136)	$\mathbf{4.45 \times 10^{-334}}$	5.11×10^{-324}	5.11×10^{-324}	5.11×10^{-324}	5.17×10^{-324}
n=28	6.96×10^{-385} (58)	$\mathbf{3.08 \times 10^{-412}}$	$\mathbf{4.74 \times 10^{-392}}$	$\mathbf{4.74 \times 10^{-392}}$	6.97×10^{-385}	6.97×10^{-385}
n=30	4.90×10^{-431} (47)	4.90×10^{-431}	4.90×10^{-431}	4.90×10^{-431}	4.90×10^{-431}	4.90×10^{-431}
n=32	3.45×10^{-474} (18)	$\mathbf{2.63 \times 10^{-519}}$	3.45×10^{-474}	3.46×10^{-474}	3.45×10^{-474}	3.45×10^{-474}

TABLE 7.7 – Constante de normalisation calculée ainsi que le $r_{\max}$ observé pour la méthode R_1 dans le cas cores periphery.

En termes de résultats obtenus, la structure cores periphery est la structure plus difficile parmi les six structures utilisées pour simuler les matrices de distances. Nous observons que dans cette structure de distances, les rangs maximaux trouvés augmentent jusqu'à atteindre 136 pour $n = 26$, cela induit des temps de calcul qui peuvent attendre à leur tour 379 secondes pour le rounding R_1 soit plus de 6 minutes pour ce calcul.
Nous constatons aussi que cette augmentation du rang n'a pas nécessairement d'influence sur les résultats. En effet, le calcul de la constante de normalisation donnent des résultats précis quand nous appliquons une compression par rounding avec les paramètres R_3 et R_4 pour $n = 24$ (resp. $n = 26$) où

	R_1	R_2	R_3	R_4	R_5	R_6
n=8	0.81	0.00	0.01	0.02	0.07	0.23
n=10	6.28	0.01	0.03	0.04	0.33	2.72
n=12	0.88	0.01	0.05	0.07	0.30	0.42
n=14	1.15	0.02	0.09	0.11	0.54	0.59
n=16	2	0.02	0.13	0.17	0.70	0.84
n=18	3.57	0.03	0.12	0.27	1.33	2.09
n=20	79.80	0.04	0.25	0.36	3.46	31.90
n=22	79.20	0.05	0.31	0.53	4.36	35.80
n=24	302	0.06	0.41	0.67	5.65	79
n=26	350	0.07	0.54	0.79	7.08	74.70
n=28	379	0.08	0.48	1.02	5.52	93.10
n=30	11.40	0.09	0.34	1.19	1.97	4.28
n=32	1.73	0.12	0.43	1.50	0.77	0.77

TABLE 7.8 – Temps de calcul en secondes pour W dans le cas cores periphery.

les rangs sont de 135 (resp. 136) tandis que, alors que ces roudings donnent des résultats très éloignés du calcul de référence pour $n = 16$ où $r_{\max} = 43$. En ce qui concerne les autres méthodes de calcul, nous remarquons que R_5 donne des résultats très précis à part pour $n = 8$.
De plus, avec cette méthode, nous faisons chuter le temps de calcul de 379 secondes à 5.52 secondes pour $n = 28$ contre 93.10 secondes (1 minute 30 seconde) pour la méthode R_6.

7.3.4.2 Réitération des calculs

Dans le cadre des expériences précédentes, nous avons fixé α et Λ et Q. De ce fait, la constante de normalisation W ne dépend plus que de la matrice distance D qui est générée aléatoirement et peut conduire entre 2 matrices différentes à de fortes variations sur W et les $r_{\max}$ observés lors des itérations de l'algorithme 1. Nous avons remarqué que pour certaines expériences, il y a de grandes différences entre la constante de normalisation calculée avec un ϵ et celles calculées avec ϵ et r_{max}. Afin de visualiser cela, nous générons 50 matrices de distance par structure de distance pour deux nombres d'individus $n = 20$ et $n = 30$ et nous comparons les résultats obtenus pour différents choix du rounding avec le calcul de la méthode R_1 (rounding avec précision) qui devient la référence.
Dans les deux tableaux 7.10 et 7.11, nous présenterons en ligne les structures de distances et en colonnes les paramètres du rounding. Chaque colonne donne le nombre de fois où le calcul avec la méthode choisie donne un résultat différent de celui obtenu avec R_1 avec une erreur relative supérieur à 0.1.
Á noter que quand l'erreur relative est supérieure à 0.1, elle est en fait de l'ordre de 1. Donc les résultats sont très loin de la référence.
NB : Pour une meilleure lisibilité des tableaux, nous mettrons en gras le nombre de fois où une méthode donne un résultat différent de notre référence. Pour les discussions et les commentaires des résultats, nous notons les paramètres du rounding de cette manière.

	$r_{max} = 9$ et ϵ	$r_{max} = 9$	ϵ et $r_{max} = 27$	$r_{max} = 50$ et ϵ	$r_{max} = 81$ et ϵ
	R_3	R_4	R_5	R_{5bis}	R_6

TABLE 7.9 – Tableau des abréviations pour différentes méthodes de rounding 2.

	Paramètre du rounding				
	R_4	R_3	R_5	R_{5bis}	R_6
associative	**1**	**1**	0	0	0
Hiérarchique	**2**	**2**	0	0	0
ordered	**2**	**2**	0	0	0
dissociative	**9**	**7**	0	0	0
cores periphery	**3**	**2**	**1**	0	0
Hiérarchique 2	**7**	**3**	0	0	0

TABLE 7.10 – Résultats pour les calculs de la constante de normalisation pour de 50 matrices de distances par structure de SBM avec un nombre d'individus $n = 20$.

	Paramètre du rounding				
	R_4	R_3	R_5	R_{5bis}	R_6
associative	**3**	**2**	0	0	0
Hiérarchique	0	0	0	0	0
Ordered	**2**	**1**	0	0	0
dissociative	**5**	**3**	0	0	0
cores periphery	**8**	**5**	0	0	0
Hiérarchique 2	**2**	**1**	0	0	0

TABLE 7.11 – Résultats pour les calculs de la constante de normalisation pour de 50 matrices de distances par structure de SBM avec un nombre d'individus $n = 30$.

Dans un premier temps, nous observons que le nombre de calculs égaux au calcul avec R_1 pour les diverses structures de distances est supérieur à 95% à part pour les cas dissociative pour $n = 20$ et cores périphérie pour $n = 30$ où la précision des calculs avec un rounding ayant comme paramètres $r_{max} = 9$ et $r_{max} = 9$ et ϵ où le nombre de calculs exacts est compris entre 80% et 90%.
Si nous comparons les tableaux 7.10 et 7.11, nous constatons que les résultats pour $n = 30$ sont légèrement meilleurs que ceux pour $n = 20$. Nous pouvons dire que plus le nombre d'individus augmente, plus l'approximation de la constante de normalisation a tendance à être proche de notre référence.
Dans un second temps, nous constatons que l'approximation avec rounding avec comme paramètres ϵ et $r_{max} = 27$ suffit largement pour obtenir un résultat précis presque tout le temps. De plus, il est plus rapide que ceux considérons comme paramètres ϵ et $r_{max} = 50$ et ϵ et $r_{max} = 81$.
En vue des résultats présentés dans cette section, nous pouvons présumer que le meilleur choix des paramètres afin d'améliorer le rapport précision du résultat et temps de calcul est le rounding avec comme paramètres $r_{max} = Q^3 = 27$ et ϵ.

7.3.4.3 Comparaison des marginales

Nous avons montré durant les sections 6.5 et 6.6 que le calcul des marginales unaires et binaires avec l'approche TT est très précis lorsque nous faisons les calculs avec une précision $\epsilon = 5 \times 10^{-2}$. Dans cette section, nous comparons les calculs des marginales avec des compressions avec un rounding de paramètre ϵ (R_1) et une autre avec un rounding de paramètres $r_{max} = Q^3$ et ϵ (R_3).
Nous commençons par choisir deux structures associative et dissociative puis $n = 18$ puis nous simulons un modèle SBM de la même manière qu'auparavant. Nous calculons ensuite les marginales et la constante de normalisation avec un rounding R_1 puis avec un rounding R_3. Enfin, dans les figures 7.7 et 7.8 nous présentons les résultats pour le cas associative avec un pairplot (expliqué dans la section 6.4.3). Dans les figures 7.9 et 7.10 nous présentons les résultats pour le cas dissociative.

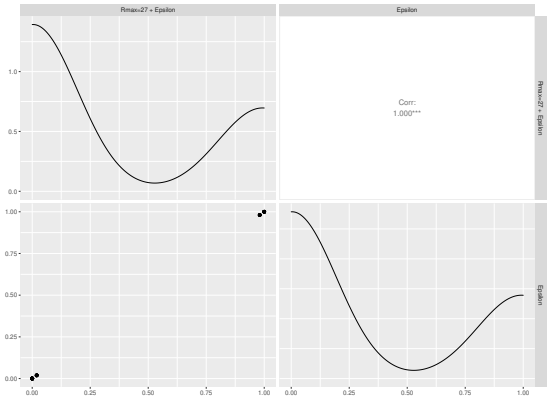


FIGURE 7.7 – Comparaison des marginales unaires calculées avec rounding R_1 et R_3 pour le cas associative.

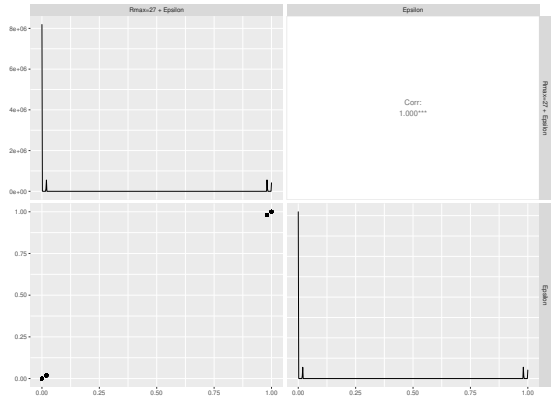


FIGURE 7.8 – Comparaison des marginales binaires calculées avec rounding R_1 et R_3 pour le cas associative.

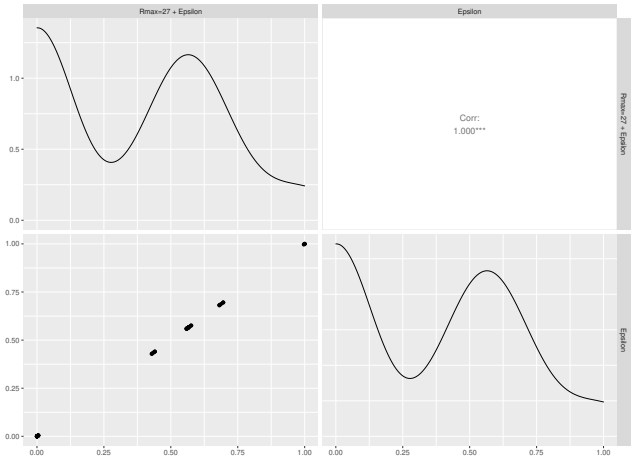


FIGURE 7.9 – Comparaison des marginales unaires avec rounding R_1 et R_3 pour le cas dissociative

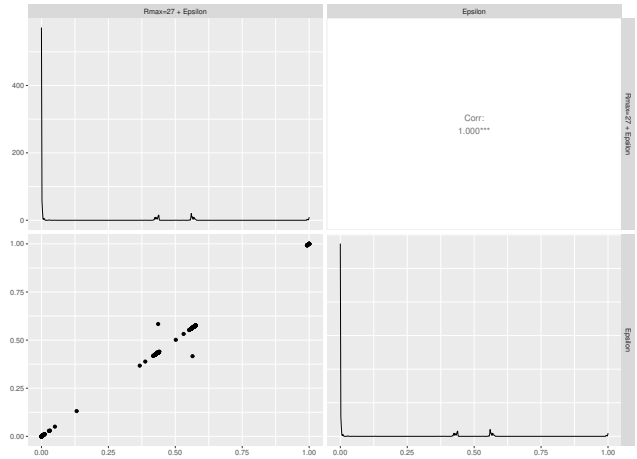


FIGURE 7.10 – Comparaison des marginales binaires avec rounding R_1 et R_3 pour le cas dissociative

Dans ces deux exemples, les calculs de la constante de normalisation donnent les mêmes résultats et les marginales sont très proches (la corrélation est de 1).
Nous présentons dans le tableau 7.12 le temps de calcul des marginales unaires et binaires pour les deux modèles. .

Paramètre du rounding	R_3	R_1
associative	35	82
dissociative	91	201

TABLE 7.12 – Temps de calcul des marginales en secondes pour un calcul avec rounding R_3 comparé à un calcul avec rounding R_1

7.3.5 Discussion

Limite du rounding avec précision . Pour un modèle SBM, le calcul avec des compressions de TT-matrices par un rounding classique (avec une précision fixée ϵ) ne permet ni de faire un calcul de la constante de normalisation si le nombre d'individus est élevé ($n > 33$) ni celui des marginales

$n > 26$.
Cela peut s'expliquer par le fait qu'utiliser le rounding classique permet de contrôler la précision ϵ mais ne permet pas de contrôler les rangs des cores durant les calculs intermédiaires, ce qui induit a des stockages de cores de rangs élevés, ce qui soit augmente les temps de calcul soit rend les calculs impossibles.
Pour palier à cette difficulté, nous avons changé les paramètres en entrées du rounding ce qui modifie la ligne 4 de l'algorithme 1 ainsi :

$$\mathbf{W} = \text{rounding}(\mathbf{W}, \epsilon, r_{max})$$

Influence des paramètres du rounding À partir des différents tableaux 7.7, 7.8, 7.5, 7.6, 7.10, 7.11 et ceux présentés dans l'annexe A, nous pouvons dire que pour les six structures de distances choisies, à part cores periphery 7.8 et dissociative A.2, le meilleur choix des paramètres du rounding en termes de temps de calcul est celui prenant $r_{max} = 9$ et $\epsilon = 5 \times 10^{-2}$ étant donné qu'il donne des résultats assez précis et le deuxième meilleur temps de calcul. Cependant, nous remarquons qu'il peut donner parfois des résultats peu précis pour certains calculs de la constante de normalisation comme nous l'avons présenté dans les tableaux 7.10 et 7.11.
Ainsi, afin d'avoir le meilleur rounding en termes de rapport précision temps de calcul pour les six structures, il est plus adéquat d'utiliser un $r_{max} = 27$ et $\epsilon = 5 \times 10^{-2}$ qui permet par exemple de faire le calcul de la constante de normalisation avec une excellente précision avec un temps de calcul assez faible (5.52 secondes au lieu de 379 pour un calcul avec une précision ϵ).

Choix optimal des paramètres du rounding A travers les différents résultats que nous avons présentés, nous avons remarqué que plus le r_{max} augmente plus la précision des calculs est meilleure jusqu'à atteindre un point de saturation quand $r_{max} = Q^3$.
Le meilleur choix en termes de rapport temps de calcul/précision est de prendre un $\epsilon = 0.05$ et $r_{max} = Q^3$.

7.3.6 Conclusion

Durant cette section, nous avons présenté comment évoluent les rangs dans les différentes itérations du calcul des marginales et de la constante de normalisation. Nous avons ensuite expliqué comment limiter cette évolution par l'intermédiaire d'une compression par l'algorithme classique du rounding puis comment changer les paramètres de cet algorithme pour limiter le stockage et accélérer les calculs. Enfin, nous avons présenté des résultats expérimentaux de comparaison de plusieurs calculs avec différents paramètres en entrées sur différentes structures de distances.
Nous avons ajouté cette procédure dans la méthode de calcul présentée dans la section 5.3 et nous l'avons associée à l'homothétie que nous avons présenté dans la section 7.2 permis de faire les calculs de la constante de normalisation et des marginales pour un modèle avec un nombre d'individus égale à 44.
Cependant, nous avons présenté dans les tableaux 7.10 et 7.11 que dans certains cas, les calculs de la constante de normalisation donne des résultats peu précis.

7.4 Réduction des dimensions de TT-matrices de rang 1 par fusion des cores

7.4.1 Motivation

Nous avons expliqué précédemment que pour un modèle SBM avec n individus, nous manipulons des TT-matrices avec $m = \frac{n(n-1)}{2}$ cores. Cela a plusieurs répercussions sur les calculs.

Tout d'abord, au niveau de la complexité, plus le nombre de cores augmente plus la longueur de la boucle sur les dimensions et le temps des calculs augmentent, que ce soit lors de la construction des $\mathbf{B}_i$ (formule (5.2.33)), pour les produits des TT-matrices TT-vecteurs $\mathbf{B}_i$ et pour le rounding nécessaires au calcul de la constante des marginales et de la constate de normalisation.
De plus, dans la section 5.3, nous montrons que le calcul des marginales binaires nécessite un nombre quadratique ($\mathcal{O}(Q^2n^2)$, équation 5.3.16) de produits TT-matrice-TT-matrice.
Chacun de ces produits de TT-matrices impose $\mathcal{O}(n^2)$ factorisations LQ et décompositions SVD, donc, le calcul de la constante de normalisation (resp. le calcul les marginales binaires) nécessite $\mathcal{O}(n^3)$ (resp. $\mathcal{O}(Q^2n^4)$) factorisations LQ et décompositions SVD.

D'autre part, au niveau de l'implémentation, la majorité des bibliothèques qui implémentent ces factorisations sont optimisées pour des matrices de grandes tailles. Or, les cores des TT-matrices $\mathbf{A}_i[z_i]$ sont de très petites tailles comme nous avons expliqués dans la formule (5.2.24).
De plus, nous avons implémenté les calculs en utilisant la librairie `ttpy` (Tenseur Train (TT) toolbox en Python) développée en 2015 par Oseledets [OTZ11]. La bibliothèque impose une limite de 1024 pour la dimension des tenseurs au format TT. Le nombre d'individus du modèle SBM est donc limité à $n = 45$ pour pouvoir construire les TT-matrices $\mathbf{A}_i[z_i]$.
Afin de résoudre ce problème, nous proposons plusieurs méthodes de fusion des cores pour réduire les dimensions des TT-matrices $\mathbf{A}_i[z_i]$.

7.4.2 Explication de la fusion des cores

7.4.2.1 Réduction du nombre des cores d'une TT-matrice de rang 1

Pour expliquer la procédure de fusion des cores, nous commençons par l'illustrer sur un exemple où nous considérons une TT-matrice $\mathbf{A}$ d'ordre 6 ayant pour dimensions $(n_0, n_1, \ldots, n_6) \times (m_0, m_1, \ldots, m_6)$ et dont les cores sont de rangs $(r_0, r_1, \ldots, r_6)$. Les éléments de $\mathbf{A}$ peuvent s'exprimer :

$$\mathbf{A}[i_1, \ldots, i_6 ; j_1, \ldots, j_6] = \mathbf{G}_1(i_1, j_1).\mathbf{G}_2(i_2, j_2).\ldots.\mathbf{G}_6(i_6, j_6) \tag{7.4.1}$$

avec $\mathbf{G}_k \in \mathbb{R}^{r_{k-1} \times n_k \times m_k \times r_k}$ le k^e core de $\mathbf{A}$.
Si $\mathbf{A}$ est de TT-rang 1, les rangs de ces TT-cores sont toujours égaux à 1,
En développant les produits matriciels, $\mathbf{A}$ s'écrit :

$$\mathbf{A}[i_1, \ldots, i_6 ; j_1, \ldots, j_6] = \sum_{\alpha_1=1}^{r_1} \ldots \sum_{\alpha_5=1}^{r_5} \mathbf{G}_1(1, i_1, j_1, \alpha_1)\mathbf{G}_2(\alpha_1, i_2, j_2, \alpha_2) \ldots \mathbf{G}_6(\alpha_5, i_6, j_6, 1)$$

qui se simplifie en un produit de scalaire

$$\mathbf{A}[i_1, \ldots, i_6 ; j_1, \ldots, j_6] = \mathbf{G}_1(1, i_1, j_1, 1)\mathbf{G}_2(1, i_2, j_2, 1) \ldots \mathbf{G}_6(1, i_6, j_6, 1) \tag{7.4.2}$$

Supposons que nous souhaitons fusionner les cores de $\mathbf{A}$ par groupe de deux, c'est-à-dire trouver une TT-matrice $\mathbf{B}$ d'ordre 3 qui permet d'exprimer les éléments de $\mathbf{A}$:

$$\mathbf{B}[\overline{i_1 i_2}, \overline{i_3 i_4}, \overline{i_5 i_6} ; \overline{j_1 j_2}, \overline{j_3 j_4}, \overline{j_5 j_6}] = \mathbf{A}[i_1, \ldots, i_6 ; j_1, \ldots, j_6] \tag{7.4.3}$$

Nous utilisons la notion de multiplexage présentée dans la section 4.4.1 et la fonction ρ définie dans la formule (4.4.3) pour exprimer les indices de $\mathbf{B}$ de cette manière :

$k_1 = \rho((i_1, i_2)) = \overline{i_1 i_2}$, $k_2 = \rho((i_3, i_4)) = \overline{i_3 i_4}$, $k_3 = \rho((i_5, i_6)) = \overline{i_5 i_6}$, $l_1 = \rho((j_1, j_2)) = \overline{j_1 j_2}$, $l_2 = \rho((j_3, j_4)) = \overline{j_3 j_4}$, $l_3 = \rho((j_5, j_6)) = \overline{j_5 j_6}$
En développant $\mathbf{B}$ nous obtenons :

$$\mathbf{B}[k_1, k_2, k_3 ; l_1, l_2, l_3] = \sum_{\alpha'_1=1}^{r'_1} \sum_{\alpha'_2=1}^{r'_2} \mathbf{C}_1(1, k_1, l_1, \alpha'_1)\mathbf{C}_2(\alpha'_1, k_2, l_2, \alpha'_2)\mathbf{C}_3(\alpha'_2, k_3, l_3, 1) \tag{7.4.4}$$

où les TT-cores $\mathbf{C}$, de $\mathbf{B}$ s'expriment par produit de Kronecker des TT-cores initiaux $\mathbf{G}$:

$$\mathbf{C}_1(k_1,l_1) = \mathbf{G}_1(i_1,j_1) \otimes_k \mathbf{G}_2(i_2,j_2), \qquad \mathbf{C}_2(k_2,l_2) = \mathbf{G}_3(i_3,j_3) \otimes_k \mathbf{G}_4(i_4,j_4),$$
$$\mathbf{C}_3(k_3,l_3) = \mathbf{G}_5(i_5,j_5) \otimes_k \mathbf{G}_6(i_6,j_6) \tag{7.4.5}$$

Puisque les cores de la TT-matrice $\mathbf{B}$ sont calculés par produits de Kronecker des cores de la TT-matrice $\mathbf{A}$. Alors, ils vérifient les propriétés suivantes :

Proposition 9. *Les TT-matrices* $\mathbf{B}$ *sont de TT-rang* 1 *:*

$$rang(\mathbf{C}_1) = rang(\mathbf{C}_2) = rang(\mathbf{C}_3) = 1$$

Démonstration. Comme les TT-matrices $\mathbf{A}$ sont de TT-rang 1 alors :

$$\mathbf{G}_1(i_1,j_1).\mathbf{G}_2(i_2,j_2).\ldots.\mathbf{G}_6(i_6,j_6) \in \mathbb{R}$$

Les cores de $\mathbf{B}$ sont calculés par produits de Kronecker des cores de $\mathbf{A}$. Nous utilisons la propriété de Kronecker qui stipule que pour deux matrices A et B nous avons : $rang(A \otimes_k B) = rang(A) \times rang(B)$. Donc,

$$\begin{cases} \text{rang}(\mathbf{C}_1(k_1,l_1)) & = & \text{rang}(\mathbf{G}_1(i_1,j_1)) & \times & \text{rang}(\mathbf{G}_2(i_2,j_2)) = 1 \\ \text{rang}(\mathbf{C}_2(k_2,l_2)) & = & \text{rang}(\mathbf{G}_3(i_3,j_3)) & \times & \text{rang}(\mathbf{G}_4(i_4,j_4)) = 1 \\ \text{rang}(\mathbf{C}_1(k_3,l_3)) & = & \text{rang}(\mathbf{G}_5(i_5,j_5)) & \times & \text{rang}(\mathbf{G}_6(i_6,j_6)) = 1 \end{cases} \tag{7.4.6}$$

Donc les TT-matrices $\mathbf{B}$ sont de TT-rang 1. □

Cette procédure peut facilement se généraliser pour des TT-matrices de TT-rang 1 d'ordre quelconque.

Proposition 10 (Réduction des dimensions TT-matrices)**.** *Soit* $\mathbf{A}$ *une TT-matrice d'ordre* m *et de TT-rang* 1*. Nous pouvons réduire l'ordre de cette TT-matrice vers un ordre* $d < m$ *.*

$$\mathbf{B}[k_1,\ldots,k_d\,;l_1,\ldots,l_d] = \mathbf{A}[i_1,\ldots,i_m\,;j_1,\ldots,j_m] \tag{7.4.7}$$

Les cores de la TT-matrices $\mathbf{B}$ *est fait par produit de Kronecker des cores de* $\mathbf{A}$ *comme présenté dans la formule* (7.4.5)*.*

Proposition 11 (Réduction des dimensions de TT-matrices de rang 1)**.** *Soit* $\mathbf{A}$ *une TT-matrice de rang* 1*, alors, si nous réduisons ses dimensions, la TT-matrice résultante est de rang* 1*.*

La démonstration de cette propriété vient de la généralisation de la formule (7.4.6).

Dans le cadre des modèles graphiques, la procédure de Novikov [NROV14] génère des TT-matrices $\mathbf{A}_i[z_i]$ sont des TT-matrices de TT-rang 1. Donc la procédure de fusion peut être appliqué sans aucune réserve.
Comme les TT-matrices $\mathbf{B}_i$ sont construites par somme des TT-matrices $\mathbf{A}_i[z_i]$. Alors, les rangs des cores sont $r_1 = \ldots = r_{d-1} = Q$ et $r_0 = r_d = 1$. Nous pouvons donc appliquer la procédure de fusion sur les TT-matrices $\mathbf{B}_i$ de manière à vérifier la condition (2.). Dans la suite de cette section, nous allons présenter cette procédure sur des TT-matrices de TT-rang 1.

7.4.3 Fusion des cores de la TT-matrice $\mathbf{A}_i[z_i]$

L'idée derrière cette méthode de fusion consiste à transformer la TT-matrice $\mathbf{A}_i[z_i]$ d'ordre m en une TT-matrice d'ordre d avec $d < m$.
Pour cela, nous découpons l'intervalle $[\![1,m]\!]$ en d sous paquets $(P_k)_{k\in[\![1,d]\!]}$. Puis, nous construisons les nouveaux cores comme produits de Kronecker des cores d'un paquet. Pour la TT-matrice qui s'exprime :

$$\mathbf{A}_i[z_i][i_1,\ldots,i_m;j_1,\ldots,j_m] = \mathbf{G}_i^1[z_i](i_1,j_1)\mathbf{G}_i^2[z_i](i_2,j_2)\ldots\mathbf{G}_i^m[z_i](i_m,j_m) \tag{7.4.8}$$

avec $\mathbf{G}_i^\ell[z_i] \in \mathbb{R}^{1\times n_\ell \times m_\ell \times 1}$ le ℓ^e core de $\mathbf{A}_i[z_i]$, la transformation permet de la réécrire :

$$\mathbf{A}_i[z_i][l_1,\dots,l_d;k_1,\dots,k_d] = \mathbf{C}_i^1[z_i](l_1,k_1)\mathbf{C}_i^2[z_i](l_2,k_2)\dots\mathbf{G}_i^{d+1}[z_i](l_d,k_d) \tag{7.4.9}$$

avec,

$$\forall f \in [\![1,d+1]\!], \quad \mathbf{C}_i^f[z_i] = \bigotimes_{\ell\in P_f}{}_k\mathbf{G}_i^\ell[z_i] \tag{7.4.10}$$

avec : $\mathbf{C}_i^f[z_i] \in \mathbb{R}^{1\times\left(\Pi_{\ell\in P_f} n_\ell\right)\times\left(\Pi_{\ell\in P_f} m_\ell\right)\times 1}$

Définition 2 (Volume d'un core)**.** *Le volume d'un core est défini comme le logarithme du produit des dimensions des cores.*
Si nous considérons un core $\mathbf{G}_i^\ell[z_i] \in \mathbb{R}^{r_{\ell-1}\times n_\ell\times m_\ell\times r_\ell}$, son volume s'exprime :

$$V_i^\ell = \log(r_{\ell-1}\times m_\ell \times m_\ell \times r_\ell) = \log(r_{\ell-1}) + \log(n_\ell) + \log(m_\ell) + \log(r_\ell) \tag{7.4.11}$$

Alors $\forall \ell \in [\![1,m]\!]$ le volume de chaque cores de $\mathbf{A}_i[z_i]$ s'exprime,

$$V_i^\ell = \log(n_\ell) + \log(m_\ell) \tag{7.4.12}$$

7.4.3.1 Fusion avec un nombre uniforme de cores

La fusion avec un nombre uniforme de cores consiste en un découpage uniforme de l'intervalle $[\![1,m]\!]$ en d intervalles égaux et un dernier intervalle avec le reste.
Soit p la taille d'un paquet, alors le nombre de paquets est lié à m par la relation suivante :

$$m = dp + s \qquad \text{avec} \qquad s \in [\![0,p-1]\!] \tag{7.4.13}$$

Dans ce cas, la nouvelle TT-matrice contiendra un nombre de cores égale à $d = \lceil \frac{m}{p} \rceil + 1$ si $s \neq 0$ et $d = \lceil \frac{m}{p} \rceil$ si $s = 0$.
Dans la figure 7.11, nous indiquons la distribution du nombre de cores, P_f que contiendra chaque nouveau core $\mathbf{C}_i^k[z_i]$ dans le cadre $d = 6$.

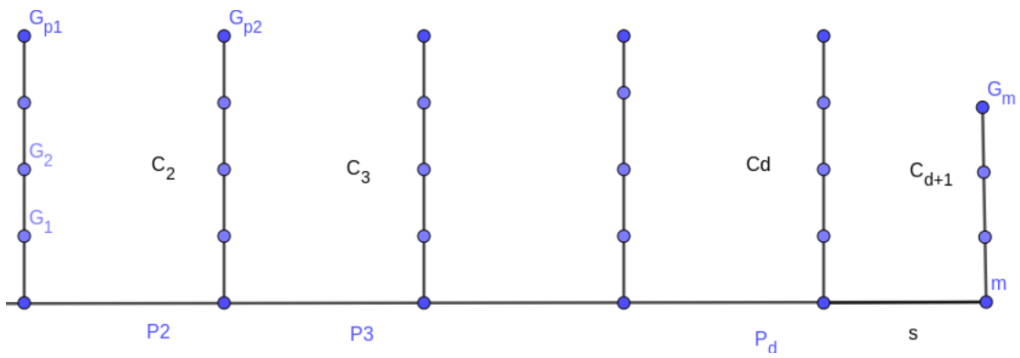


FIGURE 7.11 – Répartition uniforme en 6 paquets des m cores

Procédure de fusion : Considérons $\mathbf{A}_i[z_i]$ une TT-matrice d'ordre m et de TT-rang 1. Soit $p \in [\![1,m]\!]$.
La procédure de la fusion avec un nombre uniforme de paquet consiste à :

1. Définir l'ensemble des paquets P_f

$$\forall f \in [\![1,d]\!], \qquad P_f = [(f-1)p+1, \min(fp,m)] \tag{7.4.14}$$

où d est le nouvel ordre de la TT-matrice $A_i[z_i]$.

2. Chaque nouveau core est formé par le produit de Kronecker de p cores, le dernier core est formé par produit de Kronecker des s derniers cores.
   Construire les cores $\mathbf{C}_i^f[z_i]$ de la TT-matrice $\mathbf{A}_i[z_i]$ d'ordre d qui s'exprime :

$$\mathbf{A}_i[z_i][l_1,\dots,l_d;k_1,\dots,k_d] = \mathbf{C}_i^1[z_i](l_1,k_1)\mathbf{C}_i^2[z_i](l_2,k_2)\dots\mathbf{G}_i^d[z_i](l_d,k_d)\mathbf{G}_i^{d+1}[z_i](l_{d+1},k_{d+1}) \tag{7.4.15}$$

   où $\mathbf{C}_i^f[z_i]$ s'exprime selon la formule (7.4.10)

NB : Le choix de p doit être fait de sorte que la taille des cores ne soit pas trop grande. Pour ce choix, nous préconisons un entier $p \in [\![2,7]\!]$ si $Q=3$. Si nous considérons $p=8$, la taille des cores peut atteindre dans le pire des cas une taille de 6561×6561 ce qui est difficilement manipulable car cela demandera beaucoup de mémoire.

Avantages et inconvénients de cette méthode : L'avantage majeur de cette méthode est qu'elle nous permet de passer d'une TT-matrice d'ordre m à une TT-matrice d'ordre d plus petit avec une procédure très simple.
L'inconvénient majeur est que les cores qui composent les TT-matrices ne sont pas tous de mêmes tailles. Donc si nous appliquons cette procédure, nous aurons une mauvaise distribution des volumes des cores. En effet, certains cores auront un grand volume tandis que d'autres auront un volume nul.
Nous illustrons les difficultés qui apparaissent dans cette méthode par un exemple où nous considérons un modèle SBM [1] avec $n=60$ (soit un nombre de liens $m=1770$). Nous traçons dans le graphique 7.12 le nombre de cores identités de tailles $1 \times Q \times Q \times 1$ en bleu et le nombre de cores de tailles $1 \times 1 \times 1 \times 1$ en orange.

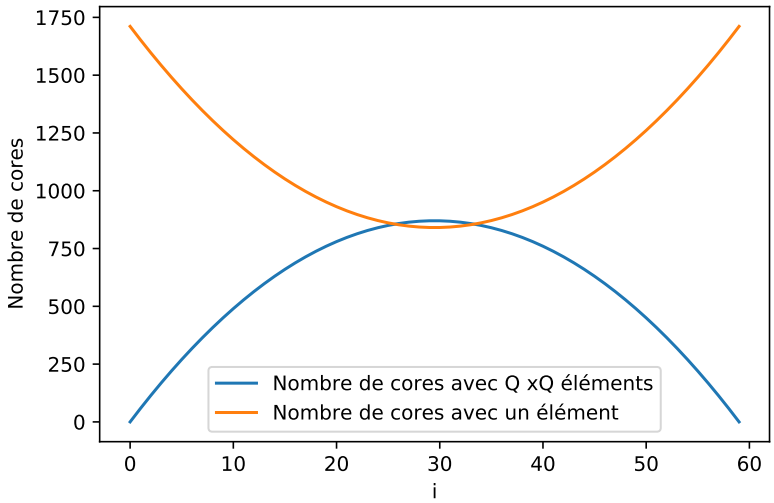


FIGURE 7.12 – Évolution du nombre de cores par rapport à l'évolution de i l'indice des TT-matrices $\mathbf{A}_i[z_i]$ pour un modèle SBM avec $n=60$ ($m=1770$)

Nous constatons dans ce graphique que les cores de grandes tailles sont concentrés dans les TT-matrices $\mathbf{A}_i[z_i]$ où i est proche de $\frac{n}{2}$. Ce nombre atteint $870 \sim \frac{m}{2}$ pour la TT-matrice $\mathbf{A}_{\frac{n}{2}}[z_{\frac{n}{2}}]$.
Nous remarquons aussi le grand déséquilibre pour les premières et dernières TT-matrices $\mathbf{A}_i[z_i]$ où nous avons très peu de cores identités mais beaucoup de cores $1 \times 1 \times 1 \times 1$. Le maximum vaux 1711 et est atteint pour $\mathbf{A}_1[z_1]$ et $\mathbf{A}_n[z_n]$.
Cela engendre des problèmes lorsque nous faisons une fusion avec un nombre uniforme de cores car nous serons amené à fusionner soit des cores de très petites tailles soit des cores de très grandes tailles.

1. A noté que les tailles des cores des TT-matrices $A_i[z_i]$ ne dépendent pas de la structure de distance choisie

7.4.3.2 Fusion avec volumes uniformes des cores

Nous avons expliqué durant la procédure de fusion dans la sous-section 7.4.3.1 que nous devons choisir judicieusement le nombre de cores à fusionner afin d'éviter les limites de dépassement de mémoire. Dans cette sous-section, nous allons désormais nous intéresser à la taille des cores des TT-matrices $\mathbf{A}_i[z_i]$.
Nous avons précisé les tailles des cores dans la formule (5.2.24). À partir de cette formule, nous pouvons pressentir que pour un modèle SBM avec n individus, la TT-matrice qui contiendra les plus grands cores est la TT-matrice $\mathbf{A}_{\frac{n}{2}}$.

Nous nous sommes intéressés au nombre exact des cores de dimensions $1 \times Q \times Q \times 1$. Pour cela, nous proposons la propriété 12.

Proposition 12. *Soit $n \in \mathbb{N}$, la TT-matrice $\mathbf{A}_i[z_i], i = 1, \dots, n$ est constituée de :*

- *(P_1) $i-1$ cores de tailles $1 \times Q \times 1 \times 1$*
- *(P_2) $n-i$ cores de tailles $1 \times 1 \times Q \times 1$*
- *(P_3) $(i-1)(n-i)$ cores de tailles $1 \times Q \times Q \times 1$*
- *(P_4) $\frac{(n-1)(n-2)}{2} - (i-1)(n-i)$ cores de tailles $1 \times 1 \times 1 \times 1$*

Cette proposition se démontre de la même manière que la proposition 5.

Démonstration. Soit $n \in \mathbb{N}$, $m = \frac{n(n-1)}{2}$ et $i \in [\![1, n]\!]$. La matrice $\mathbf{A}_i[z_i]$ s'exprime :

$$\mathbf{A}_i[z_i] = \mathbf{G}_i^{1,2}[z_i] \bullet_{\alpha_1} \mathbf{G}_i^{1,3}[z_i] \bullet_{\alpha_2} \dots \bullet_{\alpha_{d-2}} \mathbf{G}_i[z_i]^{n-2,n} \bullet_{\alpha_{d-1}} \mathbf{G}_i[z_i]^{n-1,n} \tag{7.4.16}$$

$$\text{Avec,} \quad \left| \begin{array}{lll} i < k & \Rightarrow \mathbf{G}_i^{k,\ell}[z_i] = 1 & \in \mathbb{R}^{1\times1\times1\times1} \\ i = k & \Rightarrow \mathbf{G}_i^{k,\ell}[z_i] = G_i^{i,l}[z_i] & \in \mathbb{R}^{1\times1\times Q\times1} \\ k < i < \ell & \Rightarrow \mathbf{G}_i^{k,\ell}[z_i] = \mathbb{I}_{Q,Q} & \in \mathbb{R}^{1\times Q\times Q\times1} \\ i = l & \Rightarrow \mathbf{G}_i^{k,\ell}[z_i] = G_i^{k,i}[z_i] & \in \mathbb{R}^{1\times Q\times1\times1} \\ i > \ell & \Rightarrow \mathbf{G}_i^{k,\ell}[z_i] = 1 & \in \mathbb{R}^{1\times1\times1\times1} \end{array} \right. \tag{7.4.17}$$

Le nœud i apparaît dans les liens suivants :

- Par la droite : $1 \to i, 2 \to i, \dots, i-1 \to i$
- Par la gauche : $i \to i+1, i \to i+2, \dots, i \to n$

À partir de ces constats, nous pouvons déterminer les emplacements des différents cores.

- Quand i apparaît à droite d'une connexion, cela donne des cores de taille $1 \times Q \times 1 \times 1$: Nous observons que nous avons $i-1$ connexions par la droite et donc $i-1$ cores de taille $1 \times Q \times 1 \times 1$. Ce qui donne ($P_1$).
- Quand i apparaît à gauche d'une connexion, cela donne des cores de taille $1 \times 1 \times Q \times 1$: Comme nous avons $n-i$ connexions à gauche donc $n-i$ cores de taille $1 \times 1 \times Q \times 1$. D'où ($P_2$).
- Á partir des deux premières propriétés, nous avons $m - (n-1) = \frac{(n-1)(n-2)}{2}$ cores scalaires ou des identités. Nous utilisons la formule (7.4.17) pour remarquer que les cores identités sont :

$$\underbrace{\mathbf{G}_i^{1,i+1}[z_i], \mathbf{G}_i^{1,i+2}[z_i], \dots \mathbf{G}_i^{1,n}[z_i]}_{n-i}, \underbrace{\mathbf{G}_i^{2,i+1}[z_i], \mathbf{G}_i^{2,i+2}[z_i] \dots \mathbf{G}_i^{2,n}[z_i]}_{n-i}, \dots \underbrace{\mathbf{G}_i^{i-1,i+1}[z_i], \dots \mathbf{G}_i^{i-1,n}[z_i]}_{n-i}$$

  Donc, il y aurait tout simplement $(i-1)(n-i)$ cores identités. Ce qui conduit à (P_3).
- Á partir de (P_1), (P_2) et (P_2), nous avons $\frac{(n-1)(n-2)}{2} - i(n-i)$ cores de tailles $1 \times 1 \times 1 \times 1$.

□

Proposition 13. *La TT-matrice* $\mathbf{A}_{\frac{n}{2}}[z_{\frac{n}{2}}]$ *est celle qui contient le plus de de cores de tailles* $1 \times Q \times Q \times 1$.

Démonstration. Les matrices $\mathbf{A}_i[z_i]$ contiennent $(i-1)(n-i)$ cores de tailles $1 \times Q \times Q \times 1$. Soit :

$$\begin{aligned} f : [\![1,n]\!]^n &\xrightarrow{f} \mathbb{N} \\ i &\longrightarrow (i-1)(n-i) \end{aligned} \tag{7.4.18}$$

f est une fonction dérivable et sa dérivée est : $\forall i \in [\![1,n]\!] f'(i) = n - 2i$.
Alors,

$$f'(i) = 0 \Leftrightarrow i = \frac{n}{2}$$

Le maximum de f est atteint en $\frac{n}{2}$ donc la TT-matrice $\mathbf{A}_{\frac{n}{2}}[z_{\frac{n}{2}}]$ est celle qui contient le plus de de cores de tailles $1 \times Q \times Q \times 1$ □

Si nous nous basons sur le graphique 7.12 et la propriété 12, nous pouvons dire qu'une meilleure fusion des cores se ferait en fusionnant les cores de sorte à avoir des cores de mêmes volumes.
Contrairement à la fusion des cores avec un nombre uniforme de cores, la fusion des cores avec des volumes de cores uniformes nécessite un pré-traitement qui permettra de définir le nombre de cores (P_k) dans chaque paquet et donc nous n'avons pas nécessairement l'égalité des nombres cores dans chaque paquet qui vérifient $P_k \in [1,m]$ tel que $\sum_k P_k = m$. De plus, nous aurons une contrainte qui consiste à commencer par trouver le meilleur regroupement des cores selon $\mathbf{A}_{\frac{n}{2}}[z_{\frac{n}{2}}]$ puis le propager dans les autres cores.

Dans cette méthode de fusion, chaque nouveau core est composé d'un nombre de cores P_k. Dans la figure 7.13, nous présentons la distribution du nombre de cores P_k que contiendra chaque nouveau cores $\mathbf{C}_i^k[z_i]$ dans le cadre ou $d = 5$.

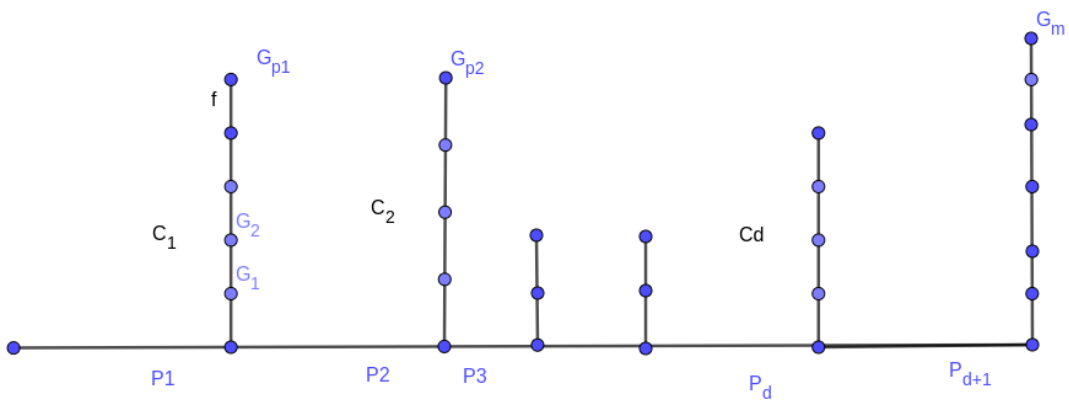


FIGURE 7.13 – Répartition des cores dans les paquets de volumes uniformes de cores

Procédure de fusion. Considérons $\mathbf{A}_i[z_i]$ une TT-matrice d'ordre m et de TT-rang égal à 1. Cette procédure se découpe en quatre phases : premièrement, nous faisons une analyse symbolique sur les facteurs afin de trouver la meilleure manière d'équilibrer les cores pour l'ensemble des $\mathbf{A}_i[z_i]$. Deuxièmement, nous faisons une phase de rééquilibrage, ensuite une phase de propagation du rééquilibrage et enfin une phase de fusion des cores par produits de kronecker.
Nous détaillons les différentes étapes de cette procédure :

1. **Phase d'initialisation.** Nous faisons la procédure de Novikov de manière symbolique afin de déterminer les volumes des cores et des TT-matrices $\mathbf{A}_i[z_i]$

(a) **Traitement symbolique des facteurs.** Construction symbolique des cores des TT-matrices $\mathbf{A}_i[z_i]$.
Nous déterminons dans un premier temps les emplacements (c'est-à-dire les indices ℓ et i) des cores de taille $1 \times r \times 1 \times 1$ et ceux de taille $1 \times 1 \times r \times 1$ obtenus après la phase de décomposition des facteurs par une SVD.
Nous déterminons ensuite les cores de taille $1 \times r \times r \times 1$ et $1 \times 1 \times 1 \times 1$ obtenus lors de la phase d'ajout des dimensions non essentielles.

(b) **Calcul des volumes des différents cores.** Nous calculons le volume V_i^ℓ de chaque core $\mathbf{G}_i^\ell[z_i]$ qui compose les différentes TT-matrices $\mathbf{A}_i[z_i]$.

(c) **Calcul des volumes des $\mathbf{A}_i[z_i]$.** Nous calculons la somme V_i^ℓ pour obtenir les volumes des TT-matrices $\mathbf{A}_i[z_i]$.

$$V^{\mathbf{A}_i} = \sum_{\ell=1}^{m} V_i^\ell$$

2. **Phase de rééquilibrage.** Nous avons remarqué que les matrices identités se concentrent dans les TT-matrices du milieu et principalement la TT-matrice $\mathbf{A}_{\frac{n}{2}}$. Nous allons utiliser $V^{\mathbf{A}_{\frac{n}{2}}}$ pour faire un premier équilibrage des cores. Cet équilibrage se fait de la manière suivante :

   (a) Choisir une contrainte $V_{\max}$, le volume maximal des cores.

   (b) Choisir $d \in [\![1, m]\!]$ tel que $d+1$ est notre nombre de cores.

   (c) Calcul de la taille moyenne optimale que devraient avoir les nouveaux corps :

$$\text{size} = \frac{V^{\mathbf{A}_{\frac{n}{2}}}}{d}$$

   (d) Regrouper les cores en d groupes de sorte à ne pas dépasser la taille moyenne optimale.

   (e) Mettre les cores restant dans un seul core puis calculer V^{d+1} leur volume total.
   Si $V^{d+1} > V_{\max}$, alors morceler ce nouveau core en autant de paquets de cores qu'il faut afin de respecter cette contrainte. Nous notons ce nombre de paquets b.

   Á la fin de cette phase, nous obtenons une liste avec les indices ℓ des cores à fusionner afin d'avoir un bon équilibrage pour la TT-matrice $\mathbf{A}_{\frac{n}{2}}[z_{\frac{n}{2}}]$.

3. **Phase de propagation du rééquilibrage sur les autres TT-matrices :** Cette phase consiste à faire une propagation du réequilibrage présenté auparavant en respectant la contrainte des cores à équilibrer.
   Pour faire cette propagation, nous parcourons les indices de 1 à n en vérifiant que si nous fusionnons les cores des autres $\mathbf{A}_i[z_i]$ nous respectons la contrainte.

4. **Construction des TT-matrices $\mathbf{A}_i[z_i]$:** Calculer les $p+b$ nouveaux cores par produit de Kronecker.

$$\forall 1 \leq k \leq d+b-1, \qquad \mathbf{C}_i^k[z_i] = \bigotimes_{\ell=(k-1)\times P_k+1}^{k\times P_{k+1}} \mathbf{G}_i^\ell[z_i] \tag{7.4.19}$$

   Á la fin de cette étape, nous obtenons les TT-matrices $\mathbf{A}_i[z_i]$:

$$\mathbf{A}_i[z_i][i_1, \ldots, i_{d+1}; j_1, \ldots, j_{d+1}] = \mathbf{C}_i^1[z_i](i_1, j_1)\mathbf{C}_i^2[z_i](i_2, j_2)\ldots\mathbf{G}_i^{d+b}[z_i](i_{d+b}, j_{d+b}) \tag{7.4.20}$$

7.4.4 Résultats

7.4.4.1 Influence de la méthode de la fusion sur les volumes des cores

Nous nous intéresserons aux logarithmes des volumes des cores des TT-matrices. Pour cela, nous considérons un modèle SBM avec $n = 15$ et une structure de distance associative et nous traçons la heatmap des volumes des cores pour différentes fusions.

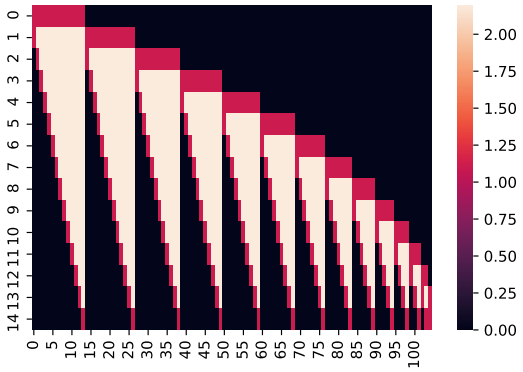


FIGURE 7.14 – Volumes des cores des $\mathbf{A}_i[z_i]$ sans fusion ($n = 15$ et $m = 105$)

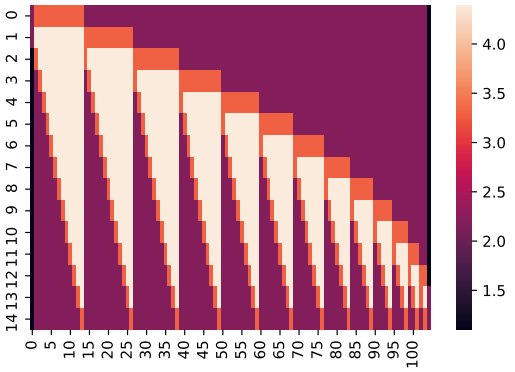


FIGURE 7.15 – Volumes des cores des $\mathbf{B}_i$ sans fusion ($n = 15$ et $m = 105$)

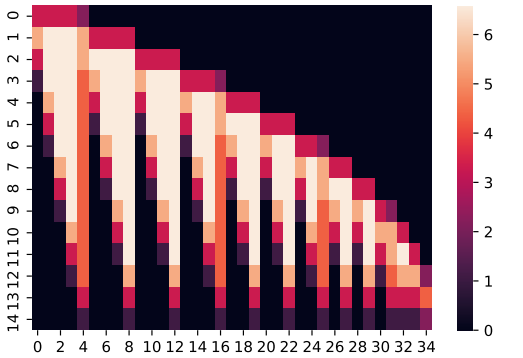


FIGURE 7.16 – Volumes des cores des $\mathbf{A}_i[z_i]$ pour la fusion par un nombre uniforme de cores ($n = 15$ et $d = 35$)

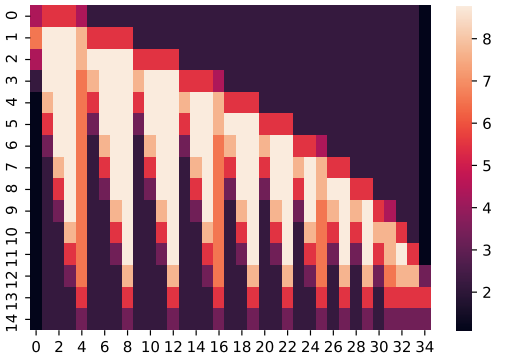


FIGURE 7.17 – Volumes des cores des $\mathbf{B}_i$ pour la fusion par un nombre uniforme de cores ($n = 15$ et $d = 35$)

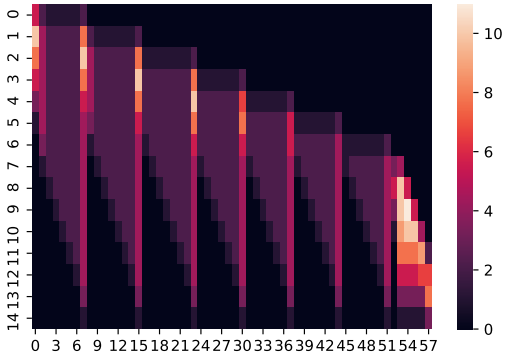


FIGURE 7.18 – Volumes des cores des $\mathbf{A}_i[z_i]$ avec fusion par un volume uniforme de cores ($n = 15$ et $d = 58$)

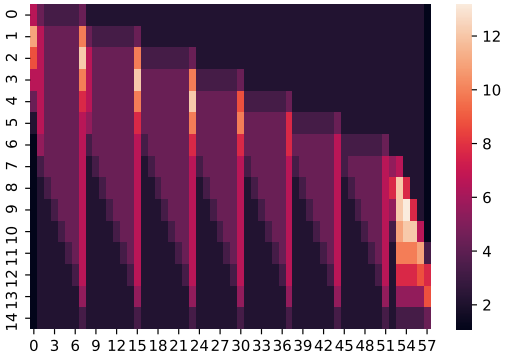


FIGURE 7.19 – Volumes des cores des $\mathbf{B}_i$ avec fusion par un volume uniforme de cores ($n = 15$ et $d = 58$)

Les trois figures 7.14, 7.16 et 7.18 représentent les heatmaps des volumes des cores des TT-matrices $\mathbf{A}_i[z_i]$. Les trois figures 7.15, 7.17 et 7.19 représentent les heatmaps des volumes des cores des TT-matrices $\mathbf{B}_i$.
Pour chacune de ces figures, nous traçons en ligne les indices i des TT-matrices et en colonnes les indices de leurs cores.

— Dans les graphes 7.14 et 7.15 nous traçons les volumes des TT-matrices sans fusion le nombre de cores est 105.

— Dans les graphes 7.16 et 7.17 nous traçons les volumes des TT-matrices avec fusion avec un nombre uniforme de cores par paquet de trois, le nombre de cores est 35.

— Dans les graphes 7.18 et 7.19 nous traçons les volumes des TT-matrices avec fusion avec un volume uniforme de cores par paquet de trois, le nombre de cores est 58.

Nous constatons que les mêmes motifs qui apparaissent dans heatmaps des cores $\mathbf{A}_i[z_i]$ se retrouvent dans les heatmaps des cores $\mathbf{B}_i$. Nous allons juste faire une comparaison des figures 7.14, 7.16 et 7.18.
Tout d'abord, nous nous intéressons aux similarités entre les trois heatmaps. Nous remarquons qu'une grande partie des motifs sont partagés par les trois figures (les pixels noirs). Cela vient du fait que les TT-matrices $\mathbf{A}_i[z_i]$ pour $i = 1,\ldots,5$ et $i = 12,\ldots 15$ contiennent beaucoup de cores de volumes nuls. Cela vient du fait que seuls $(i-1)(n-i)$ cores sont de tailles $1 \times Q \times Q \times 1$. Dans le cas où $i = 1$ ou 15, nous n'aurons aucun core de taille $1 \times Q \times Q \times 1$. Dans le cas où $i = 2$ ou 14, nous aurons 14 cores de cette taille.
Nous observons aussi que les heatmaps 7.14 et 7.16 sont très similaires. Cela est dû au fait que comme nous fusionnons les cores de manière consécutive par paquet de 3, les mêmes patterns de volumes des cores se retrouvent sans fusion et avec fusion avec nombre uniforme de cores.
Nous observons aussi que pour la fusion de cores en paquets uniformes, il y a beaucoup de cores avec des volumes proches de 6.5, tandis que pour la fusion par volumes uniformes des cores, très peu de cores ont un grand volume : seuls les six derniers cores des TT-matrices $\mathbf{A}_i[z_i]$ ont des cores avec de très grandes tailles (le volume maximal est de 10.4).
Á partir de ces figures, nous pouvons dire que la fusion avec volumes uniformes des cores nous donne une meilleure distribution des volumes des cores.

7.4.4.2 Diminution des rangs des cores lors du calcul de W

Nous avons remarqué que lors du calcul de la constante de normalisation sans fusion et avec fusion, il y a une différence pour les TT-rangs observés lors des calculs intermédiaires, et cela, quelque soit la structure de distances choisie. Afin d'illustrer ce phénomène, nous avons fait trois expériences.
Nous définissons un nombre de classes $Q = 3$ et les proportions d'individus dans les classes $\alpha = \left(\frac{1}{3}, \quad \frac{1}{3}, \quad \frac{1}{3}\right)$ et une structure de distances Λ associative (pour 7.20) et cores periphery (pour les figures 7.21 et 7.22).
Nous choisissons trois nombres d'individus : $n = 15$, $n = 33$ et $n = 50$. Puis, pour les deux graphes 7.20 et 7.21, nous appliquons la procédure de Novikov à la fois sans fusion des cores (courbe orange), avec une fusion avec nombres un uniforme de cores par paquets de trois (courbe bleue) puis, avec un volume uniforme des cores (courbe verte). Ensuite, nous calculons la constante de normalisation pour chaque découpage avec une compression via l'algorithme du rounding avec comme paramètres $\epsilon = 5 \times 10^{-2}$.
Enfin, dans le graphique 7.22, nous appliquons la procédure de Novikov avec fusion des cores d'abord, avec nombre uniforme de cores (par paquet de trois) puis avec un volume uniforme de cores étant donné que la procédure sans fusion ne peut être appliquée. De plus, nous compressons les résultats intermédiaires via un rounding ayant pour paramètres $\epsilon = 5 \times 10^{-2}$ et $r_{\max} = 27$.

Dans les graphiques 7.20, 7.21 et 7.22, nous traçons l'évolution des TT-rangs maximaux des cores obtenus après chaque itération du calcul de la constante de normalisation.

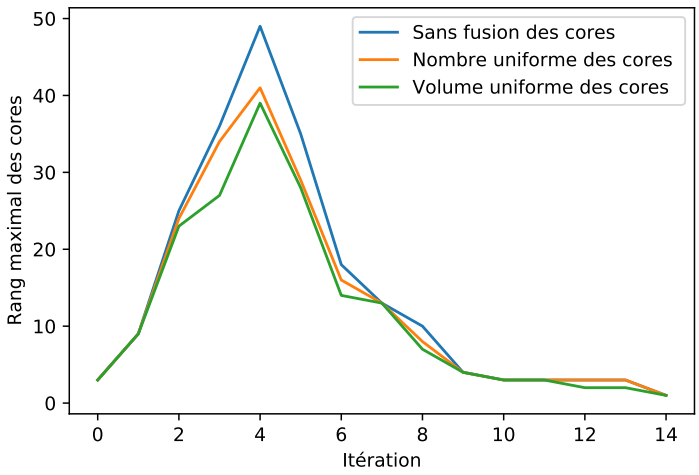


FIGURE 7.20 – Évolution des rangs des cores en fonction des itérations de l'algorithme 1 pour les 3 procédures ($n = 15$, $\epsilon = 5 \times 10^{-2}$ et $r_{\max} = 27$)

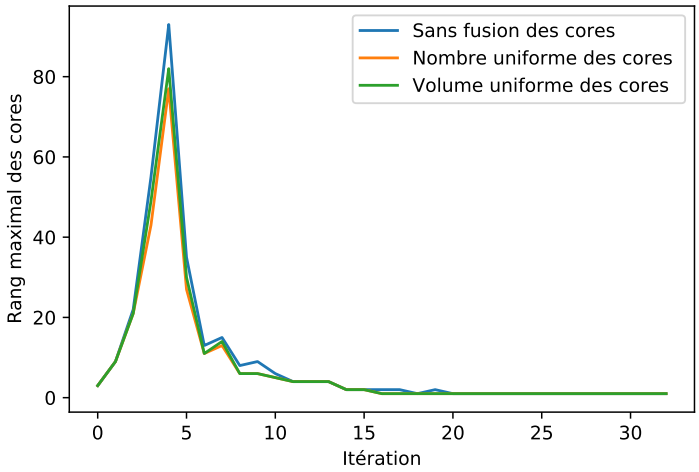


FIGURE 7.21 – Évolution des rangs des cores en fonction des itérations de l'algorithme 1 pour les 3 procédures ($n = 33$, $\epsilon = 5 \times 10^{-2}$ et $r_{\max} = 27$)

Dans les deux graphiques 7.20 et 7.21, nous constatons que la fusion des cores permet de diminuer le rang lors du calcul de la constante de normalisation. En effet, dans la figure 7.20 le rang maximal des cores observé atteint 48 lors de l'itération 4 de l'algorithme 1 pour le calcul sans fusion, tandis qu'il n'est que de 40 (resp. 39) pour le calcul avec fusion avec nombre de cores uniforme (resp. pour le calcul avec fusion avec volumes uniformes).
Concernant le temps de calcul pour la constante de normalisation, il est relativement le même pour les trois méthodes. Nous exprimons les temps de calculs dans le tableau 7.4.4.2.
Dans la figure 7.22, les rangs maximaux des cores des TT-matrices par itération de calcul par les deux méthodes sont les mêmes quelle que soit la procédure de fusion adoptée.

7.4.4.3 Comparaison des constantes de normalisations obtenues par les trois méthodes

Protocole. Nous définissons un nombre de classes $Q = 3$ et les proportions d'individus dans les classes $\alpha = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ et une structure de distances Λ associative.
Puis, nous faisons varier le nombre d'individus n entre 12 et 42 par pas de 3 et nous appliquons la

Méthode	Sans fusion	Fusion avec nombres uniformes	Fusion avec volumes uniformes
$n = 15$	0.83 s	0.79 s	0.81 s
$n = 33$	3.35 s	3.42 s	3.40 s

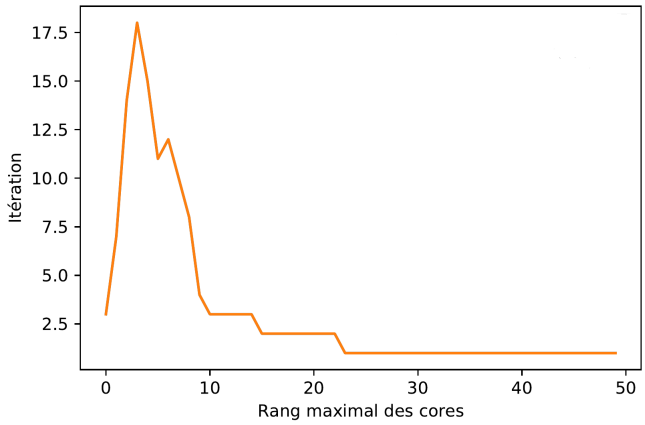


FIGURE 7.22 – Évolution des rangs des cores en fonction des itérations de l'algorithme 1 en changeant les paramètres du rounding ($n = 50$, $\epsilon = 5 \times 10^{-2}$ et $r_{\max} = 27$)

procédure de Novikov sur le modèle à la fois sans fusion des cores, avec une fusion avec un nombre uniforme de cores (par paquets de trois) et avec un volume uniforme de cores.
Enfin, nous calculons de la constante de normalisation avec une compression via un rounding avec comme paramètres $\epsilon = 5 \times 10^{-2}$ et $r_{\max} = 27$.

Résultats : Nous présentons les résultats dans le tableau 7.13 avec en ligne le nombre d'individus et en colonnes la constante de normalisation. Nous observons que les erreurs relatives entre les différentes procédures de fusion sont très faibles voir nulles, le maximum de ces erreurs est atteint en $n = 27$ où il vaut 0.008.
Les erreurs relatives entre les calculs sans fusion et les deux méthodes de fusion et le calcul sans fusion est très faible lui aussi. Le maximum est de 0.08 et est atteint pour $n = 27$.

n	Sans fusion	Fusion avec nombres uniformes (paquet de 5 cores)	Fusion avec volumes uniformes
12	9.35×10^{-71}	9.35×10^{-71}	9.35×10^{-71}
15	8.04×10^{-112}	7.95×10^{-112}	7.95×10^{-112}
18	3.77×10^{-163}	3.77×10^{-163}	3.77×10^{-163}
21	2.92×10^{-219}	2.92×10^{-219}	2.92×10^{-219}
24	1.53×10^{-281}	1.53×10^{-281}	1.53×10^{-281}
27	3.49×10^{-359}	3.76×10^{-359}	3.73×10^{-359}
30	4.40×10^{-459}	4.40×10^{-459}	4.40×10^{-459}
33	7.81×10^{-565}	7.81×10^{-565}	7.81×10^{-565}
36	1.55×10^{-673}	1.54×10^{-673}	1.55×10^{-673}
39	1.08×10^{-775}	1.08×10^{-775}	1.08×10^{-775}
42	1.06×10^{-915}	1.06×10^{-915}	1.06×10^{-915}

TABLE 7.13 – Tableau présentant les résultats de la constante de normalisation avec et sans fusion

7.4.5 Discussion

Durant cette section, nous nous sommes intéressés à la fusion des cores, nous avons expliqué comment réduire les dimensions des TT-matrices grâce à une procédure de fusion des cores par produits de Kronecker. Nous avons expliqué deux procédures de fusion, la première avec un nombre uniforme de cores et une deuxième avec un volume uniforme de cores.

Résumé des résultats Durant la sous-section 7.4.4.1, nous avons présenté l'influence de la méthode de fusion des cores adoptées sur les nouveaux cores résultants. Nous avons montré que la fusion avec des volumes uniformes des cores conduits à une meilleure distribution des nombres d'éléments dans les cores dans les TT-matrices $\mathbf{A}_i[z_i]$ et $\mathbf{B}_i$.
Durant la sous-section 7.4.4.2, nous nous sommes intéressés au calcul de la constante de normalisation W par l'intermédiaire de la fusion des cores. Dans les résultats, nous avons remarqué que ce calcul avec fusion permet de diminuer les rangs des TT-matrices à chaque itération de l'algorithme 1.
Enfin, durant la section 7.4.4.3, nous nous sommes intéressé aux résultats de la constante de normalisation obtenus avec fusion et sans fusions des cores. Nous avons expliqué que les différences entre les différents résultats obtenus sont très faibles.

Temps de calcul. Nous avons remarqué que le temps de calcul est plus faible lorsque nous faisons la procédure sans fusion avec rounding à précision prescrite. Ce temps de calcul devient quasiment identique pour les trois méthodes lorsque le calcul se fait avec une précision et un rang max prescrits. Cependant, comme précisé lors de la sous-section 7.3.6 nous ne pouvons pas faire les calculs, que ce soit des marginales ou de la constante de normalisation pour un nombre d'individus supérieur à 44, grâce à cette procédure, nous pouvons maintenant le faire.

Passage à l'échelle Avec la procédure de fusion des cores, nous arrivons à faire les calculs des marginales et de la constante de normalisation pour des nombres d'individus relativement plus élevés. En effet, nous avons étendu la procédure de Novikov pour un nombre d'individus jusqu'à $n = 66$. Soit un modèle graphique complet avec un nombre de liens $m = 2145$.

7.4.6 Conclusion

Dans cette section, nous avons proposé deux méthodes de réduction des dimensions des TT-matrices $\mathbf{A}_i[z_i]$ utilisées dans l'approche de Novikov [NROV14]. La première consiste à faire une fusion avec un nombre uniforme de cores et la seconde avec un volume uniforme de cores.
Nous avons ensuite présenté les résultats pour chacune de ces méthodes de fusion, à partir de ces résultats, nous pouvons dire que ses deux méthodes de fusion sont très intéressantes étant donné qu'elles fournissent des résultats très précis, un rang plus faible et un temps de calcul quasiment identique que la procédure sans fusion.

7.5 Conclusion du chapitre

Durant ce chapitre, nous nous sommes interressés à trois limitations que nous avons rencontrées lors de la déclinaison de la procédure de Novikov sur des SBM que nous avons détaillé durant l'introduction. Nous résumons les procédures adoptées pour résoudre ces limitations :

Éviter les problèmes des petites valeurs. Nous avons présenté durant la section 7.2 comment faire une homothétie pour agrandir les facteurs binaires $\psi'_{i,j}$ à l'entrée de la procédure de Novikov pour éviter des valeurs nulles pour la constante de normalisation et des marginales. Nous avons ensuite expliqué quelle constante choisir pour faire ces calculs.
Grâce à cette procédure, nous avons réussi à calculer la constante de normalisation pour des SBM avec

un nombre d'individus $n = 33$ et de calculer des marginales pour des SBM avec un nombre d'individus $n = 26$.

Explosion des rangs des cores. Nous avons expliqué durant l'introduction que chaque produit de TT-matrices augmente le rang de la TT-matrice résultante et que sans rounding sur la constante de normalisation W lors de chaque itération augmentera jusqu'à atteindre Q^n alors que W est un scalaire. Nous avons expliqué durant la section 7.3, que le rounding avec une précision ϵ ne stabilise pas ce rang et qu'il faudrait modifier les paramètres de cet algorithme en le faisant avec $r_{\max}$ et ϵ. Grâce à cette procédure, nous avons réussi à accélérer les calculs.
En associant cette procédure à l'homothétie, nous avons réussi à faire les calculs de la constante de normalisation et des marginales pour un modèle avec nombre d'individus égale à 44.

Réduction du nombre de cores. Nous avons présenté dans la section 7.4 deux procédures de réduction de l'ordre des TT-matrices $\mathbf{A}_i[z_i]$ et $\mathbf{B}_i$. Une première est basée sur une fusion avec un nombre uniforme de cores et une deuxième, consiste à faire une fusion avec un volume uniforme des cores. Nous avons ensuite expliqué les avantages et les inconvénients de chaque approche.
Nous avons conclu cette section par une présentation des résultats que nous avons décomposé en trois parties, influence de la méthode sur la distribution des volumes des cores, la diminution des rangs des produits de TT-matrices TT-vecteurs lors du calcul de W et enfin, nous avons comparé les résultats de la constante de normalisation obtenus avec et sans fusion.
Á partir des résultats présentés, cette procédure, combinée aux deux autres, nous permettent d'étendre la procédure de Novikov pour des SBM avec un nombre d'individus égale à $n = 66$ soit un modèle graphique avec un nombre de liens égal à $m = 2145$.

Chapitre 8

Conclusion et perspectives

Table des matières

8.1 Rappel des contributions

Nous nous sommes intéressés au calcul des marginales binaires des SBM pondérés par un tableau de distances par une approche de type Tensor Train et nous avons présenté les contributions suivantes.

Dans le chapitre 3 et en annexe B.2, nous avons réalisé un clustering de matrices de distances de séquences de marqueurs d'intérêt taxonomique (barcodes) d'arbres guyanais. Nous avons montré les similitudes et complémentarités entre les classifications obtenues par SBM et par les méthodes plus classiques de CAH. Nous avons aussi montré que la classification SBM permet de détecter, par le biais de la distribution estimée des distances intra-classe, des situations où les classifications moléculaires peuvent être en désaccord avec la classification botanique.

Nous avons présenté dans le chapitre 5 une écriture exacte de la loi jointe d'un SBM conditionnellement à des distances observées comme un tenseur au format TT, sans approximation. Cela mène naturellement à la séparation des variables dans la loi jointe, qui permet le calcul des marginales. Nous avons expliqué les matrices intervenant dans le calcul des marginales puis décrit le processus de tensorisation (quantisation) de ces matrices qui permet de les écrire sous la forme de TT-matrices ainsi que la méthode d'implémentation des calculs des marginales.
Nous avons ensuite présenté un algorithme de type programmation dynamique pour le calcul des marginales binaires par une mutualisation des calculs ainsi que sa complexité.

Dans le chapitre 6, nous avons comparé l'approche TT pour le calcul des marginales unaires et binaires d'un SBM avec trois autres méthodes présentes dans la littérature (énumération, approximation champ moyen, échantillonneur de Gibbs). Nous avons montré dans nos résultats que l'approche TT est stable vis-à-vis de la variation du nombre d'individus et de la structure de distance. De plus, elle permet de calculer les marginales dans un temps relativement raisonnable et avec une bonne précision comparée à l'approche de Gibbs.
Cette approche soulève des difficultés numériques redoutables dès que le nombre d'individus est significatif (contrairement à l'approche par le champ moyen par exemple).

Dans le chapitre 7, nous avons étudié les verrous numériques qui surviennent lors du passage à l'échelle de l'approche TT puis nous avons présenté une procédure opérationnelle qui intègre quelques solutions à ces verrous. Premièrement, les valeurs des marginales non normalisées et de la constante de normalisation sont très petites, nous avons proposé une procédure d'homothétie pour lever ce verrou. Un deuxième problème vient de l'explosion des rangs des cores, nous avons proposé un changement de paramètres dans la compression par l'algorithme du rounding.
Un troisième verrou que nous avons rencontré est que la dimension des TT-matrices intermédiaires $\mathbf{A}_i[z_i]$ et $\mathbf{B}_i$ est quadratique par rapport au nombre d'individus. Nous avons proposé une procédure de réduction des dimensions de TT-matrices $\mathbf{A}_i[z_i]$ par fusion de ses cores.
Grâce à ces trois procédures, nous avons étendu la procédure de Novikov pour des SBM avec des nombres d'individus pouvant atteindre à $n = 66$.

Ce domaine numérique reste cependant en grande partie inexploré et mal compris.

8.2 Perspectives

8.2.1 Perspectives numériques

Lors de l'application de la procédure de Novikov [NROV14] pour le calcul des marginales des SBM, nous nous sommes confrontés à des limitations numériques quand le nombre d'individus augmente.
Afin d'y remédier, nous avons proposé trois procédures qui nous ont permis de traiter des problèmes de plus grande taille. Nous avons ainsi résolu le problème des petites valeurs. Cependant, en dépit

de ces procédures, nous sommes toujours confrontés à des limitations liées à l'évolution des rangs des cores et leur nombre durant les calculs des marginales ou de la constante de normalisation des modèles.

Nous avons essayé de lever le premier verrou par le biais d'un changement de paramètres du rounding. Nous avons réussi à faire des calculs plus rapidement, mais nous ne pouvons pas contrôler la précision (les calculs du rounding à rang prescrit sont plus rapides qu'avec une précision prescrite, mais au détriment justement de la précision). Une des perspectives est de repenser un autre algorithme de compression de TT-matrices ou de TT-vecteurs. Al Daas & al. ont proposé une librairie [DBB20] pour faires des opérations parallélisées au format TT. Dans un autre article [DBC+21], Al Daas & al. ont proposé un nouvel algorithme pour faire le rounding. Une deuxième possibilité pour éviter l'explosion du rang des cores consiste à remplacer la SVD exacte pour la séparation des variables par une approximation de rang $r < Q$ (où Q est le nombre de classe). Si $r = 1$ on se retrouve avec une approximation de type champ moyen.

Le deuxième verrou auquel nous avons été confrontés est l'augmentation du nombre de cores qui composent les TT-matrices $\mathbf{A}_i[z_i]$ et $\mathbf{B}_i$ de manière quadratique par rapport aux nombres d'individus. Nous avons essayé de résoudre ce problème par une procédure de fusion des cores des TT-matrices $\mathbf{A}_i[z_i]$ par produit de kronecker. Grâce à cette procédure, nous avons réussi à traiter des problèmes de plus grande taille, cependant nous n'avons pas réussi à dépasser le cap des SBM avec un nombre d'individus $n = 66$.
Une perspective possible pour dépasser cette limite consiste à trouver une méthode de fusion des cores plus optimale que celles abordées dans le chapitre 7.

8.2.2 Réduction du temps de calcul

Une première perspective de ce travail serait de réduire les temps de calcul nécessaires pour le calcul des marginales grâce à la parallélisation, avec deux possibilités. En effet, la plupart des calculs sont indépendants.
Premièrement, il est possible de distribuer les calculs des feuilles de l'arbre présenté dans la section 5.3 sur plusieurs nœuds d'un cluster. Pour expliquer une méthode possible pour faire cette parallélisation, nous reprenons la formule de récurrence selon le parcours de l'arbre des feuilles à la racine :

$$\begin{cases} \mathbf{G_{i,i}} = \mathbf{B}_i & \forall i \in [\![1, n-1]\!] \\ \mathbf{G_{i,j+1}} = \mathbf{G_{i,j}} \times \mathbf{B}_{j+1} & \forall j \in [\![i+1, n]\!] \end{cases} \tag{8.2.1}$$

Nous reprenons la figure 5.2 présentée dans la section 5.3 (où le nombre d'individus est $n = 5$) en coloriant de la même couleur les ensembles de calculs indépendants.
Dans la figure 8.1, la construction de cet arbre nécessite 10 calculs. Ces derniers peuvent être rassemblés en quatre ensembles indépendants : le calcul $\mathbf{G}_{12}$, $\mathbf{G}_{13}$, $\mathbf{G}_{14}$ et $\mathbf{G}_{15}$ (noir), celui de $\mathbf{G}_{23}, \mathbf{G}_{24}$, $\mathbf{G}_{2F5}$ (orange), celui de $\mathbf{G}_{34}$, $\mathbf{G}_{35}$ (bleu) et celui de $\mathbf{G}_{45}$ (vert). Nous remarquons que chaque ensemble ne dépend que de $\mathbf{B}_1, \ldots, \mathbf{B}_5$. Nous pouvons donc distribuer la totalité des calculs de cet arbre sur 4 nœuds d'un cluster, chaque nœud correspond à une couleur : 4 calculs pour le premier, 3 pour le second, 2 pour le troisième et un pour le dernier. Cette méthode peut se généraliser pour la distribution des calculs des feuilles d'un SBM avec un nombre d'individus $n \in \mathbb{N}$ quelconque sur $n-2$ nœuds d'un cluster de calcul. Pour le faire nous devons procéder de cette manière : pour chaque $i \in [\![1, n-1]\!]$, faire la boucle pour le calcul de $\forall j \in [\![i+1, n]\!]$, $\quad \mathbf{G_{i,j}}$ sur un nouveau nœud du cluster.

Une deuxième phase de parallélisation peut être faite au niveau du calcul des marginales. En effet, pour un modèle SBM nous devenons calculer $\frac{n(n-1)Q^2}{2}$. Chaque calcul d'une marginale est indépendant des autres. Il est donc facile de distribuer les calculs des marginales sur plusieurs nœuds d'un cluster de calcul.

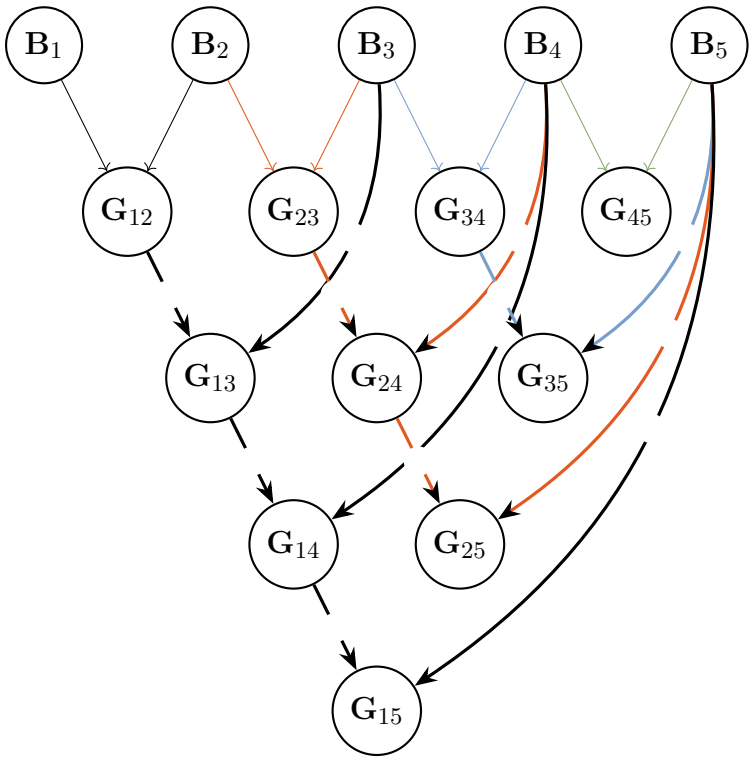


FIGURE 8.1 – Décomposition des calculs des parties centrales de l'expression des marginales binaires

8.2.3 Proposition d'un algorithme TT-EM pour l'estimation d'un SBM

Nous avons présenté durant la section 2.3 l'algorithme EM [DLR77] où chaque itération se décompose en deux étapes : une étape de calcul d'espérance (E-step) qui nécessite le calcul de toutes les marginales binaires conditionnelles et une étape de maximisation (M-step).
Nous pouvons proposer une alternative à cet algorithme via une approche TT : l'algorithme TT-EM prendra comme entrées : D matrice des distances, Q nombre de classes, n nombre d'individus, ϵ précision du rounding, τ critère d'arrêt de l'algorithme. Il retournera les classes des individus, α^{opt} et Λ^{opt}, les paramètres du modèle SBM.

L'avantage de cet algorithme est que l'approche TT permet de calculer les marginales avec une bonne précision comme nous avons expliqué dans le chapitre 6. Cependant, il reste limité comparé à des algorithmes concurrents comme le VEM [DPR08] utilisés dans diverses applications comme [TBC17, APF22]. En effet, même s'il repose sur une moins bonne approximation des marginales, le VEM donne des résultats de très bonnes qualités en terme d'estimation des paramètres d'un SBM, avec un temps de calcul raisonnable.

8.2.4 Extension de la procédure de Novikov à d'autres modèles graphiques

Nous avons travaillé sur un modèle SBM, dont tous les facteurs sont binaires et pour le calcul des marginales binaires. Chacun de ces trois choix peut être étendu comme perspective ouverte par ce travail. Nous les détaillons ci-dessous.
Appliquer l'approche pour le calcul des marginales binaires d'un modèle graphique dont les facteurs sont binaires au plus, comme le modèle d'Ising par exemple [BRU67, Nis09]. Notre approche peut se transposer "nec varietur" à de tels modèles. L'approche sera plus simple, car le modèle SBM est le cas le pire quant au nombre de facteurs (en $n(n-1)/2$ alors que pour le modèle d'Ising il est en $2n$).
L'approche de Novikov pour le calcul de la constante de normalisation ou des marginales unaires s'applique à tout modèle graphique. Si les facteurs sont d-naires (avec $d > 2$), on perd l'exactitude de

la décomposition TT : on remplace l'exactitude de la SVD pour la séparation des variables des facteurs binaires présentée dans la section 5.2.3 par une approximation par TT-SVD sur les facteurs d-naires si $d > 2$. On obtient alors une meilleure approximation par un tenseur en format TT de chaque facteur. Au-delà de ce changement, la suite des calculs se décline sans changement, et notre approche pour le calcul des marginales binaires peut se décliner également. On calcule les TT-matrices intermédiaires $\mathbf{A}_i[z_i]$ et $\mathbf{B}_i$ par l'approche de Novikov puis on utilise l'algorithme présenté dans la section 5.3 pour le calcul des marginales des binaires. Celles-ci s'expriment :

$$\forall i,j \in [1,n]^2, p_{i,j}(z_i,z_j) = \underbrace{\mathbf{B}_1 \times \ldots \times \mathbf{B}_{i-1}}_{\mathbf{L}_i} \mathbf{A}_i[z_i] \underbrace{\mathbf{B}_{i+1} \times \ldots \times \mathbf{B}_{j-1}}_{\mathbf{C}_{i,j}} \mathbf{A}_j[z_j] \underbrace{\mathbf{B}_{j+1} \times \ldots \times \mathbf{B}_n}_{\mathbf{R}_i} \tag{8.2.2}$$

Cette approche peut également se décliner pour le calcul des marginales k-naires (où $k > 2$) d'un modèle graphique avec des facteurs d-naires. Le calcul des marginales ternaires par exemple s'exprime : $\forall i,j,k \in [\![1,n]\!]^3, \quad p_{i,j,k}(z_i,z_j,z_k)$

$$= \underbrace{\mathbf{B}_1 \times \ldots \times \mathbf{B}_{i-1}}_{\mathbf{L}_i} \mathbf{A}_i[z_i] \underbrace{\mathbf{B}_{i+1} \times \ldots \times \mathbf{B}_{j-1}}_{\mathbf{C}_{i,j}} \mathbf{A}_j[z_j] \underbrace{\mathbf{B}_{j+1} \times \ldots \times \mathbf{B}_{k-1}}_{\mathbf{C}_{j,k}} \mathbf{A}_k[z_k] \underbrace{\mathbf{B}_{k+1} \times \ldots \times \mathbf{B}_n}_{\mathbf{R}_k} \tag{8.2.3}$$

La formule (8.2.3) peut être généralisée pour des marginales k-naires en rajoutant plusieurs parties centrales de cette manière : Pour $\forall i_1, i_2, \ldots i_k \in [\![1,n]\!]^k$

$$p_{i_1,i_2,\ldots i_k}(z_{i_1}, z_{i_2}, \ldots z_{i_k}) = \mathbf{L}_{i_1}\mathbf{A}_{i_1}[z_{i_1}]\mathbf{C}_{i_1,i_2}\mathbf{A}_{i_2}[z_{i_2}]\ldots\mathbf{A}_{i_{k-1}}[z_{i_{k-1}}]\mathbf{C}_{i_{k-1},i_k}\mathbf{A}_{i_k}[z_{i_k}]\mathbf{R}_{i_k} \tag{8.2.4}$$

Les matrices $\mathbf{C}_{i_{k-1},i_k}$ peuvent être calculées via la mutualisation des calculs de la même manière que pour des modèles graphiques avec des facteurs binaires.
Á partir de cela, la procédure présentée dans le chapitre 5 peut être transposée pour le calcul des marginales de n'importe quel modèle graphique.

Appendices

Annexe A

Complément des résultats sur les méthodes de passage à l'échelle de l'approche TT

Dans cette annexe, nous complétons les résultats présentés dans la sous-section résultats de la section 3 du chapitre. Nous présentons les résultats du calcul de la constante de normalisation pour les quatre structures Hiérarchique, ordonnée, dissociative et hiérarchique 2 que nous avons omis.
Nous rappelons le tableau des paramètres du rounding utilisé pour les discussions et les commentaires des résultats.

	ϵ	$r_{max} = 3$ et ϵ	$r_{max} = 9$ et ϵ	$r_{max} = 9$	$r_{max} = 27$ et ϵ	$r_{max} = 81$ et ϵ
	R_1	R_2	R_3	R_4	R_5	R_6

Table A.1 – Tableau des abréviations pour différentes méthodes de rounding.

Hiérarchique

	R_1 ($r_{\max}$)	R_2	R_3	R_4	R_5	R_6
n=8	1.35×10^{-25}(8)	1.35×10^{-25}	1.33×10^{-25}	1.35×10^{-25}	1.35×10^{-25}	1.35×10^{-25}
n=10	3.97×10^{-43} (21)	$\mathbf{4.43 \times 10^{-85}}$	3.96×10^{-43}	3.98×10^{-43}	3.97×10^{-43}	3.97×10^{-43}
n=12	4.39×10^{-73} (13)	4.39×10^{-73}	4.39×10^{-73}	4.41×10^{-73}	4.39×10^{-73}	4.39×10^{-73}
n=14	1.63×10^{-81} (14)	$\mathbf{2.20 \times 10^{-84}}$	1.63×10^{-81}	1.63×10^{-81}	1.63×10^{-81}	1.63×10^{-81}
n=16	6.38×10^{-111} (37)	$\mathbf{2.44 \times 10^{-140}}$	6.38×10^{-111}	6.43×10^{-111}	6.38×10^{-111}	6.38×10^{-111}
n=18	4.18×10^{-137}(57)	$\mathbf{6.78 \times 10^{-188}}$	4.37×10^{-137}	4.49×10^{-137}	4.18×10^{-137}	4.18×10^{-137}
n=20	2.61×10^{-164}(59)	$\mathbf{7.13 \times 10^{-168}}$	2.25×10^{-164}	$\mathbf{9.69 \times 10^{-168}}$	2.61×10^{-164}	2.61×10^{-164}
n=22	4.34×10^{-190}(5)	4.35×10^{-190}	4.35×10^{-190}	4.34×10^{-190}	4.34×10^{-190}	4.34×10^{-190}
n=24	5.01×10^{-229}(16)	$\mathbf{1.63 \times 10^{-263}}$	5.04×10^{-229}	5.02×10^{-229}	5.01×10^{-229}	5.01×10^{-229}
n=26	3.15×10^{-285}(9)	$\mathbf{1.55 \times 10^{-408}}$	3.15×10^{-285}	3.15×10^{-285}	3.15×10^{-285}	3.15×10^{-285}
n=28	1.09×10^{-333} (100)	1.09×10^{-333}	1.09×10^{-333}	1.09×10^{-333}	1.09×10^{-333}	1.09×10^{-333}
n=30	2.64×10^{-384} (18)	2.64×10^{-384}	2.64×10^{-384}	2.64×10^{-384}	2.64×10^{-384}	2.64×10^{-384}
n=32	2.64×10^{-439}(11)	2.64×10^{-439}	2.64×10^{-439}	2.64×10^{-439}	2.64×10^{-439}	2.64×10^{-439}

Table A.2 – Temps de calcul en secondes pour la constante de normalisation dans le cas Hiérarchique

	R_1	R_2	R_3	R_4	R_5	R_6
n=8	0.22	0.01	0.01	0.02	0.11	0.10
n= 10	0.15	0.01	0.02	0.03	0.06	0.06
n=12	2.55	0.01	0.04	0.06	0.40	1.00
n=14	0.02	0.01	0.01	0.10	0.01	0.01
n=16	1.46	0.02	0.06	0.17	0.28	0.49
n=18	0.57	0.02	0.08	0.23	0.24	0.25
n=20	55.60	0.03	0.22	0.34	2.71	22.40
n=22	3.49	0.04	0.14	0.45	0.46	0.97
n=24	0.43	0.04	0.07	0.56	0.10	0.10
n=26	2.68	0.05	0.12	0.70	0.63	0.81
n=28	0.75	0.06	0.16	0.94	0.20	0.20
n=30	9.67	0.08	0.25	1.17	2.23	4.07
n=32	3.73	0.10	0.35	1.47	1.36	1.36

TABLE A.3 – Temps de calcul en secondes pour la constante de normalisation dans le cas Hiérarchique

Le calcul de la constante de normalisation fait avec le rounding R_2 donne des valeurs très éloignées de la référence que nous avons choisi R_1.
Les calculs avec R_3, R_5 et R_6 donne des résultats très proches de R_1. De plus, les temps de calculs sont très bas lorsque nous utilisons la méthode R_3, nous pouvons donc dire que ce sont les meilleurs paramètres en termes de compromis temps de calcul précision pour cette structure. Dans ce qui va suivre, nous éliminons la méthode R_2 de la discussion étant donné qu'elle fournit des résultats peu précis.

Ordonnée

	R_1 ($r_{\max}$)	R_2	R_3	R_4	R_5	R_6
8	1.06×10^{-26} (7)	1.04×10^{-26}	1.04×10^{-26}	1.06×10^{-26}	1.06×10^{-26}	1.06×10^{-26}
10	1.27×10^{-49} (15)	$\mathbf{7.78 \times 10^{-50}}$	1.27×10^{-49}	1.26×10^{-49}	1.27×10^{-49}	1.27×10^{-49}
12	1.89×10^{-66} (12)	1.89×10^{-66}	1.89×10^{-66}	1.89×10^{-66}	1.89×10^{-66}	1.89×10^{-66}
14	6.47×10^{-93} (12)	6.47×10^{-93}	6.47×10^{-93}	6.47×10^{-93}	6.47×10^{-93}	6.47×10^{-93}
16	1.39×10^{-122} (17)	$\mathbf{5.87 \times 10^{-123}}$	1.39×10^{-122}	1.39×10^{-122}	1.39×10^{-122}	1.39×10^{-122}
18	2.44×10^{-159} (22)	2.44×10^{-159}	2.44×10^{-159}	2.45×10^{-159}	2.44×10^{-159}	2.44×10^{-159}
20	1.30×10^{-191} (22)	$\mathbf{2.00 \times 10^{-200}}$	1.30×10^{-191}	1.30×10^{-191}	1.30×10^{-191}	1.30×10^{-191}
22	2.51×10^{-240} (29)	$\mathbf{1.19 \times 10^{-290}}$	2.51×10^{-240}	2.51×10^{-240}	2.51×10^{-240}	2.51×10^{-240}
24	5.82×10^{-280} (33)	$\mathbf{1.51 \times 10^{-305}}$	5.82×10^{-280}	5.82×10^{-280}	5.82×10^{-280}	5.82×10^{-280}
26	1.03×10^{-323} (15)	$\mathbf{5.31 \times 10^{-340}}$	1.03×10^{-323}	1.03×10^{-323}	1.03×10^{-323}	1.03×10^{-323}
28	4.12×10^{-383} (36)	$\mathbf{2.06 \times 10^{-383}}$	4.12×10^{-383}	4.12×10^{-383}	4.12×10^{-383}	4.12×10^{-383}
30	2.54×10^{-442} (48)	$\mathbf{7.47 \times 10^{-485}}$	2.53×10^{-442}	2.54×10^{-442}	2.54×10^{-442}	2.54×10^{-442}
32	5.6×10^{-496} (14)	5.6×10^{-496}	5.6×10^{-496}	5.6×10^{-496}	5.6×10^{-496}	5.6×10^{-496}

TABLE A.4 – Constante de normalisation calculées ainsi que les $r_{\max \text{ trouvé}}$ dans le cas ordonnés

	R_1	R_2	R_3	R_4	R_5	R_6
8	0.24	0	0.01	0.02	0.16	0.12
10	0.03	0.01	0.01	0.04	0.01	0.01
12	0.69	0.01	0.06	0.08	0.33	0.33
14	0.29	0.01	0.06	0.11	0.15	0.14
16	0.05	0.02	0.02	0.17	0.02	0.03
18	0.38	0.02	0.09	0.25	0.19	0.19
20	1.95	0.04	0.15	0.35	0.80	1.06
22	1.05	0.05	0.15	0.48	0.53	0.54
24	0.75	0.06	0.16	0.65	0.33	0.33
26	10.10	0.06	0.20	0.85	2.18	4.92
28	1.37	0.08	0.31	1.05	0.56	0.56
30	15.60	0.11	0.64	1.40	7.11	8.09
32	17.10	0.12	0.36	1.52	2.92	7.84

TABLE A.5 – Temps de calcul en secondes pour la constante de normalisation dans le cas ordonnée

Les quatre méthodes de rounding R_3, R_4, R_5 et R_6 donnent le même résultat, qui est le même que celui de la méthode de référence.
De plus, R_3 nous offre un meilleur temps de calcul que les autres méthodes.

Dissociative

	R_1 ($r_{\max}$)	R_2	R_3	R_4	R_5	R_6
n= 8	3.57×10^{-32}(12)	$\mathbf{1.68 \times 10^{-32}}$	3.57×10^{-32}	3.57×10^{-32}	3.57×10^{-32}	3.57×10^{-32}
n= 10	2.08×10^{-44} (12)	$\mathbf{5.96 \times 10^{-45}}$	2.08×10^{-44}	2.08×10^{-44}	2.08×10^{-44}	2.08×10^{-44}
n= 12	1.77×10^{-63} (12)	1.26×10^{-63}	1.77×10^{-63}	1.77×10^{-63}	1.77×10^{-63}	1.77×10^{-63}
n= 14	6.79×10^{-89} (13)	$\mathbf{8.13 \times 10^{-90}}$	6.79×10^{-89}	6.79×10^{-89}	6.79×10^{-89}	6.79×10^{-89}
n= 16	2.03×10^{-117} (14)	$\mathbf{8.42 \times 10^{-120}}$	2.03×10^{-117}	2.03×10^{-117}	2.03×10^{-117}	2.03×10^{-117}
n= 18	1.08×10^{-151} (21)	$\mathbf{8.63 \times 10^{-152}}$	$\mathbf{7.79 \times 10^{-152}}$	$\mathbf{7.79 \times 10^{-152}}$	1.08×10^{-151}	1.08×10^{-151}
n= 20	3.88×10^{-179} (44)	$\mathbf{8.61 \times 10^{-189}}$	2.18×10^{-179}	2.21×10^{-179}	3.88×10^{-179}	3.88×10^{-179}
n= 22	1.17×10^{-231} (42)	1.03×10^{-231}	1.17×10^{-231}	1.17×10^{-231}	1.17×10^{-231}	1.17×10^{-231}
n= 24	6.07×10^{-276} (30)	$\mathbf{3.51 \times 10^{-280}}$	$\mathbf{1.83 \times 10^{-278}}$	$\mathbf{1.83 \times 10^{-278}}$	6.07×10^{-276}	6.07×10^{-276}
n= 26	2.03×10^{-301} (19)	$\mathbf{4.97 \times 10^{-304}}$	2.03×10^{-301}	2.03×10^{-301}	2.03×10^{-301}	2.03×10^{-301}
n= 28	3.25×10^{-361} (50)	1.06×10^{-361}	2.07×10^{-361}	2.07×10^{-361}	3.23×10^{-361}	3.25×10^{-361}
n= 30	7.19×10^{-404} (23)	$\mathbf{1.79 \times 10^{-425}}$	7.13×10^{-404}	7.13×10^{-404}	7.19×10^{-404}	7.19×10^{-404}
n= 32	2.80×10^{-476} (36)	$\mathbf{4.99 \times 10^{-495}}$	$\mathbf{4.99 \times 10^{-495}}$	$\mathbf{4.99 \times 10^{-495}}$	2.80×10^{-476}	2.80×10^{-476}

TABLE A.6 – Constante de normalisation calculées ainsi que les $r_{\max \text{ trouvé}}$ dans le cas dissociative

	R_1	R_2	R_3	R_4	R_5	R_6
8	0.02	0	0.01	0.02	0.01	0.01
10	0.09	0.01	0.03	0.04	0.06	0.05
12	0.11	0.01	0.05	0.07	0.05	0.05
14	0.22	0.02	0.08	0.12	0.12	0.12
16	0.34	0.02	0.14	0.18	0.17	0.18
18	1.05	0.03	0.16	0.27	0.52	0.52
20	4.03	0.04	0.20	0.38	1.15	2.05
22	5.28	0.05	0.29	0.52	1.37	2.78
24	3.88	0.07	0.54	0.69	1.50	1.55
26	1.70	0.07	0.27	0.84	0.70	0.69
28	27.10	0.07	0.43	0.96	4.42	9.79
30	3.45	0.10	0.30	1.21	1.63	1.64
32	15.90	0.14	0.71	1.57	5.16	7.07

TABLE A.7 – Temps de calcul en secondes pour la constante de normalisation dans le cas dissociative

Les deux méthodes R_5 et R_6 sont les méthodes les plus précises. Cependant, R_5 est un peu plus rapide pour le calcul.

Hiérarchique 2

	R_1 ($r_{\max}$)	R_2	R_3	R_4	R_5	R_6
8	8.09×10^{-34} (19)	$\mathbf{3.25 \times 10^{-46}}$	8.09×10^{-34}	8.09×10^{-34}	8.09×10^{-34}	8.09×10^{-34}
10	2.20×10^{-51} (18)	$\mathbf{1.60 \times 10^{-51}}$	2.20×10^{-51}	2.20×10^{-51}	2.20×10^{-51}	2.20×10^{-51}
12	8.51×10^{-65} (29)	$\mathbf{5.18 \times 10^{-113}}$	8.51×10^{-65}	8.51×10^{-65}	8.51×10^{-65}	8.51×10^{-65}
14	2.13×10^{-111} (8)	$\mathbf{1.44 \times 10^{-120}}$	2.13×10^{-111}	1.44×10^{-102}	2.13×10^{-111}	2.13×10^{-111}
16	4.91×10^{-135} (4)	4.91×10^{-135}	4.91×10^{-135}	4.91×10^{-135}	4.91×10^{-135}	4.91×10^{-135}
18	2.27×10^{-175} (14)	2.27×10^{-175}	2.27×10^{-175}	2.27×10^{-175}	2.27×10^{-175}	2.27×10^{-175}
20	8.31×10^{-228} (13)	8.31×10^{-228}	8.31×10^{-228}	8.31×10^{-228}	8.31×10^{-228}	8.31×10^{-228}
22	1.36×10^{-265} (17)	$\mathbf{1.61 \times 10^{-545}}$	1.36×10^{-265}	1.36×10^{-265}	1.36×10^{-265}	1.36×10^{-265}
24	9.83×10^{-328} (46)	$\mathbf{1.60 \times 10^{-353}}$	9.83×10^{-328}	9.83×10^{-328}	9.83×10^{-328}	9.83×10^{-328}
26	1.79×10^{-369} (37)	1.79×10^{-369}	1.79×10^{-369}	1.79×10^{-369}	1.79×10^{-369}	1.79×10^{-369}
28	1.68×10^{-418} (37)	$\mathbf{6.96 \times 10^{-549}}$	1.68×10^{-418}	1.68×10^{-418}	1.68×10^{-418}	1.68×10^{-418}
30	2.05×10^{-498} (17)	$\mathbf{1.11 \times 10^{-532}}$	2.05×10^{-498}	2.05×10^{-498}	2.05×10^{-498}	2.05×10^{-498}
32	1.52×10^{-586} (21)	1.38×10^{-586}	1.52×10^{-586}	1.52×10^{-586}	1.52×10^{-586}	1.52×10^{-586}

TABLE A.8 – Constante de normalisation calculées ainsi que les $r_{\max \text{ trouvé}}$ dans le cas hiérarchique 2

	R_1	R_2	R_3	R_4	R_5	R_6
8	0.07	0	0.01	0.01	0.04	0.04
10	0.08	0.01	0.02	0.04	0.03	0.03
12	0.37	0.01	0.03	0.06	0.20	0.20
14	0.05	0.01	0.02	0.10	0.03	0.03
16	0.03	0.01	0.02	0.14	0.02	0.02
18	0.19	0.02	0.07	0.21	0.10	0.10
20	0.49	0.03	0.14	0.30	0.24	0.24
22	0.65	0.04	0.15	0.41	0.32	0.32
24	6.73	0.05	0.12	0.53	1.20	2.20
26	4.95	0.06	0.30	0.73	1.66	2.10
28	4.73	0.07	0.17	0.93	1.36	2.16
30	0.69	0.08	0.17	1.29	0.34	0.37
32	3.21	0.12	0.40	1.49	1.59	1.62

Table A.9 – Temps de calcul en secondes pour la constante de normalisation dans le cas hiérarchique 2

L'ensemble des résultats suggèrent que les six méthodes de rounding s'ordonnent par précision décroissance avec l'ordre suivant : $R_6 \backslash R_5$, R_3, R_4 et enfin R_2

Annexe B

Précision des calculs des marginales et la constante de normalisation

Nous présentons dans cette annexe un théorème pour borner une TT-matrice $\mathbf{B}$ obtenue par la compression d'une TT-matrice $\mathbf{A}$ via l'algorithme du rounding avec précision prescrite ϵ. Nous allons utiliser ce théorème pour présenter un corollaire qui porte sur l'erreur de la compression des produits de TT-matrices par l'algorithme du rounding avec précision prescrite.
Nous utiliserons ce corollaire afin de présenter la précision des marginales et de la constante de normalisation calculées par l'approche TT.

B.1 Compression d'une TT-matrice par le rounding

Théorème 14. *Soit* $\mathbf{A}$ *une TT-matrice. Soit* $\mathbf{B}$ *une approximation de* $\mathbf{A}$ *obtenue par l'algorithme rounding avec précision prescrite* ϵ*, c'est-à-dire :*

$$\mathbf{B} = \mathit{rounding}(\mathbf{A}, \epsilon)$$

alors :

$$(1-\epsilon)||\mathbf{A}|| \leq ||\mathbf{B}|| \leq (1+\epsilon)||\mathbf{A}|| \tag{B.1.1}$$

Démonstration. Soit $\mathbf{A}$ une TT-matrice, alors d'après Oseledets [Ose11], si on a $\mathbf{B} = \text{rounding}(\mathbf{A}, \epsilon)$ alors l'erreur s'exprime :

$$||\mathbf{A} - \mathbf{B}|| \leq \epsilon||\mathbf{A}|| \tag{B.1.2}$$

Comme $||.||$ est une norme, alors elle vérifie l'inégalité triangulaire. Donc,

$$||\mathbf{A}|| - ||\mathbf{B}|| \leq ||\mathbf{A} - \mathbf{B}||$$

D'après B.1.2, nous obtenons

$$||\mathbf{A}|| - ||\mathbf{B}|| \leq \epsilon||\mathbf{A}|| \tag{B.1.3}$$

Donc,

$$(1-\epsilon)||\mathbf{A}|| \leq ||\mathbf{B}|| \tag{B.1.4}$$

Nous procédons de la même manière pour avoir la borne supérieure de $\mathbf{B}$.

$$||\mathbf{A} - \mathbf{B}|| \leq \epsilon||\mathbf{A}||$$

Donc,

$$||\mathbf{B} - \mathbf{A}|| \leq \epsilon||\mathbf{A}||$$

Nous développons,

$$||\mathbf{A}|| + ||\mathbf{B} - \mathbf{A}|| \leq ||\mathbf{A}|| + \epsilon||\mathbf{A}||$$

D'après la propriété de l'inégalité triangulaire : $||\mathbf{A}+\mathbf{B}-\mathbf{A}|| \leq ||\mathbf{A}||+||\mathbf{B}-\mathbf{A}||$. Donc,

$$||\mathbf{A}+\mathbf{B}-\mathbf{A}|| \leq (1+\epsilon)||\mathbf{A}||$$

Donc,

$$||\mathbf{B}|| \leq (1+\epsilon)||\mathbf{A}|| \tag{B.1.5}$$

Enfin, nous pouvons borner la norme de l'erreur à partir des formules (B.1.5) et (B.1.4) :

$$(1-\epsilon)||\mathbf{A}|| \leq ||\mathbf{B}|| \leq (1+\epsilon)||\mathbf{A}|| \tag{B.1.6}$$

□

Corollaire 15 (Produit de TT-matrices)**.** *Soient* $\mathbf{A},\mathbf{B},\mathbf{C}$ *trois TT-matrices et* $,\epsilon \in \mathbb{R}$ *.*
Soient $\mathbf{D} = \mathbf{A}\times\mathbf{B}\times\mathbf{C}$ *et* $\hat{\mathbf{D}} = rounding(rounding(\mathbf{A}\times\mathbf{B},\epsilon)\times\mathbf{C},\epsilon)$
Alors,

$$(1-\epsilon)^2\mathbf{D} \leq \hat{\mathbf{D}} \leq (1+\epsilon)^2\mathbf{D} \tag{B.1.7}$$

La démonstration de ce corolaire vient du théorème 14. Ce corollaire peut se généraliser facilement pour un produit de $n \in \mathbb{N}$ TT-matrices.

B.2 Erreur relative dans le calcul des marginales et la constante de normalisation

Nous avons expliqué dans le chapitre 5 comment appliquer la procédure de Novikov [NROV14] pour des SBM et comment faire le calcul des marginales par produits de TT-matrices. Nous rappelons la formule exacte pour le calcul des marginales unaires d'un modèle avec n individus.

$$\mathbf{p}_i(z_i) = \frac{\mathbf{B}_1\times\ldots\times\mathbf{B}_{i-1}\mathbf{A}_i[z_i]\mathbf{B}_{j+1}\times\ldots\times\mathbf{B}_n}{W}$$

Nous avons expliqué dans la fin de ce chapitre ainsi que dans le chapitre 7 l'intérêt de faire des compressions par rounding à chaque itération. À partir de cela, le calcul d'une marginale unaire se fait :

$$\hat{\mathbf{p}}_i(z_i) = \frac{\text{rounding}(\ldots\text{rounding}(\mathbf{B}_1\times\mathbf{B}_2,\epsilon)\times\ldots\times\mathbf{B}_{i-1},\epsilon)\times\mathbf{A}_i[z_i],\epsilon)\times\mathbf{B}_{j+1},\epsilon)\times\ldots\times\mathbf{B}_n,\epsilon)}{W}$$

Si nous généralisons la formule (B.1.7), nous obtienons une de l'erreur :

$$(1-\epsilon)^{n-2}W\times||\mathbf{p}_i(z_i)|| \leq W||\hat{\mathbf{p}}_i(z_i)|| \leq (1+\epsilon)^{n-2}W\times||\mathbf{p}_i(z_i)|| \tag{B.2.1}$$

Ainsi,

$$(1-\epsilon)^{n-2}||\mathbf{p}_i(z_i)|| \leq ||\hat{\mathbf{p}}_i(z_i)|| \leq (1+\epsilon)^{n-2}||\mathbf{p}_i(z_i)|| \tag{B.2.2}$$

Alors, la formule (B.2.2) peut s'exprimer :

$$(1-\epsilon)^{n-2}||p_i(z_i)|| \leq ||\hat{p}_i(z_i)|| \leq (1+\epsilon)^{n-2}||p_i(z_i)|| \tag{B.2.3}$$

Ainsi, l'erreur relative des marginales unaires s'exprime :

$$(1-\epsilon)^{n-2} \leq \frac{||\hat{p}_i(z_i)||}{||p_i(z_i)||} \leq (1+\epsilon)^{n-2}$$

De la même manière, nous pouvons borner l'erreur des marginales binaires de cette manière :

$$(1-\epsilon)^{n-2} \leq \frac{||\hat{p}_{i,j}(z_i,z_j)||}{||p_{i,j}(z_i,z_j)||} \leq (1+\epsilon)^{n-2} \tag{B.2.4}$$

Article

Nous présentons dans cette annexe l'article "Does clustering of DNA barcodes agree with botanical classification directly at high taxonomic levels ? Trees in French Guiana as a case study" publié dans Molecular Ecology Resources.

Evaluating the adequacy between morphological-based and molecular-based inventories at high taxonomic level

Mohamed Anwar Abouabdallah[1,3], Nathalie Peyrard[2,*], and Alain Franc [1,3]

[1]BIOGECO, INRAE, Université de Bordeaux, 33612 Cestas, France

[2]Université de Toulouse, INRAE, UR MIAT, F-31320, Castanet-Tolosan, France

[3]Pleiade, EPC INRIA–INRAE–CNRS, Université de Bordeaux 33405 Talence France

[*]corresponding author: nathalie.peyrard@inrae.fr

Running title: Barcoding angiosperms directly at order level

Abstract

Characterising biodiversity is one of the big challenges for the coming decades. Most diversity has not been morphologically described and metabarcoding is taking over from morphological-based taxonomy to further develop inventories. The approaches have been cross-validated at the level of species and OTUs. However, many known species are not listed in reference databases. One path to speed up inventories using metabarcoding could be to directly identify individuals at coarser taxonomic levels. We therefore studied whether morphology-based and molecular-based approaches are in adequacy at genus, family and order levels. The comparison

was performed on a dataset of approximately 1,500 trees from a French Guianese experimental plot, well-established both botanically and molecularly. We used Agglomerative Hierarchical Clustering (with Ward, Complete and Single Linkage) and Stochastic Block Models (SBM), with two dissimilarity measures, one based on Smith-Waterman local alignment scores, and one based on kmers. The adequacy between morphological-based and molecular-based classifications can be good to very good at taxonomic levels beyond species, even though it decreases when taxonomic levels increase, or when using the tetramer-based distance. Adequacy is correlated with the entropy of morphological-based taxonomy and with the ratio of the mean within- and mean between-groups dissimilarities. Ward method globally leads to the best adequacy whereas Single Linkage can show poor behaviours. SBM provides a useful signal as to whether or not the dissimilarities are structured according to the botanical groups. We assert that it is time for scaling up the Ward and SBM methods to analyse massive datasets established by metabarcoding.

Keywords: taxonomy; barcoding; Stochastic Block Model; clustering; Ward method; French Guianan Trees

1 Introduction

Numerical taxonomy and hierarchical clustering have coevolved since the 1960s' (Cole, 1969; Sneath and Sokal, 1973). Both approaches rely on the assumption that the diversity of life for taxonomy, or patterns in distances between some items in clustering, are organized as a nested hierarchy, modelled as a tree. This approach has survived the revolution of molecular-based taxonomy (Hillis et al., 1996) and molecular phylogenies (Felsenstein, 2004; Yang, 2006), with a current revival due to barcoding (Floyd et al., 2002; Hebert et al., 2003), and, at the same time, metabarcoding (López-García et al., 2001; Sogin et al., 2006; Hajibabaei et al., 2011; Taberlet et al., 2012; Kermarrec et al., 2013). As far as morphological-based taxonomy is concerned, most of the diversity in many clades of

organisms is still unknown (Leray and Knowlton, 2015, point out that between 33% and 91% of all marine biodiversity has never been named), and much efforts has been devoted to speeding up the process of producing large inventories with metabarcoding by bypassing identified blocking points (Bik et al., 2012).

Therefore, the notion of OTU (which stands for Operational Taxonomic Unit) has been coined (Floyd et al., 2002; Blaxter et al., 2005). Such units are produced by clustering sets of barcodes by aggregation at a level assumed to be similar to the level of species in morphological-based classifications. The authors in Blaxter et al. (2005) emphasize that they are "agnostic" as to whether OTU are species or not. Identifying OTUs in an environmental sample and organising molecular diversity as the frequency of OTUs make it possible to produce molecular-based inventories at previously unparalleled speed.

A classical approach is therefore to build OTUs and to map them on reference databases that contain reference barcodes. A standard tool for mapping is BLAST (Altschul et al., 1990), but other and more sophisticated solutions exist (e.g., the use of Bayesian Phylogenetics, Munch et al., 2008). When taxonomic expertise and references exist at the species level, the adequacy between molecular and morphological-based classification can be excellent (Ji et al., 2013). It may happen that such a comparison is not feasible when morphological-based taxonomy is unknown or when only partial references exist. Leray and Knowlton (2015) report in their study that less than 12% of their OTUs matched with GenBank or BOLD. The same observation was made in White et al. (2010) regarding intestinal microbial flora. Hence, most of inventories with supervised learning are made at a grain often much coarser than the genus/species level when the reference databases on which a match is looked for are annotated at high taxonomic levels like that of phylum.

Trying to complete databases at the species level is known to be highly time-consuming. We privilege another approach here and suggest directly identifying coarse levels in taxonomy present in the inventory by clustering[1] the barcodes in a limited number of groups.

[1]In this article, the term *clustering* makes reference to any numerical method for the unsupervised

Supervised learning can then be used in a second step to annotate the groups found as taxa at deep taxonomic levels. This raises the question of adequacy between morphological-based and molecular-based taxonomy when clusters of sequences are built at a level coarser than species, like, e.g., class or order. Comparing morphological-based classifications and OTU produced by barcodes clustering has been thoroughly studied (see, e.g., White et al., 2010). Several methods have been recently designed and widely used for delineating species on the basis of barcodes (Pons et al., 2006; Fontaneto et al., 2008; Puillandre et al., 2012; Talavera et al., 2013; Zhang et al., 2013). However, to our knowledge, such a question has seldom been addressed at coarser levels like orders.

Our objective here is to study whether the clustering of barcodes in molecular-based taxonomy makes it possible to directly recover all the taxonomic levels beyond the species present in a sample, and, if so, with which tool, accuracy and robustness.

We performed this study on a well-known dataset where deep knowledge of the taxonomy is available for several orders of angiosperms. We selected flowering plants because the botanical classification is well known, both morphologically (it is organised as a nested system of different taxonomic levels as a classification system) and molecularly with the APG initiative, even if it is under continuous revision (The Angiosperm Phylogeny Group et al., 2016). The dataset itself (Caron et al., 2019) is composed of some 1500 trees from French Guiana which have been botanically identified and sequenced (chloroplastic marker trnH-psbA). An advantage is that we do not have to deal with the computational burden of treatment of metabarcoding data, and we can therefore concentrate on the analysis of adequacy. The question of the scaling of those methods which will show itself be effective on this well-known and manageable dataset, will be the object of further studies, and some leads are given in the Discussion section of this article.

It can be expected that there is not a clear answer to the degree of adequacy between the

grouping of the individuals, while the term *classification* designates the method's output, i.e. the partition of individuals into classes.

two visions of classification (morphology-based or molecular-based), especially over several taxonomic levels. There may be favourable situations where the adequacy is strong, and others where the two classifications are surprisingly quasi-independent of each other. To identify potential factors that may explain variations in adequacy in our study: (*i*) we varied the taxonomic level at which the classification is performed (order, family, genus, species), (*ii*) we used two definitions of dissimilarity between sequences; and, finally, (*iii*) we considered four numerical methods for the clustering of the molecular data.

More specifically, we first worked with 30 distinct sub-samples of the whole dataset, each sub-sample being either all the individuals of an order or of a family, and for each of them, we compared the botanical classification of the individuals at the next taxonomic level with the molecular-based classifications. In a second step, we studied whether the mean behaviour observed from these replicates is recovered when the set of individuals to be classified carries more diversity, by comparing the botanical classification of the whole dataset (into orders, families, genera and species) with the molecular-based classifications.

Regarding the computation of dissimilarity between sequences of unequal length, the Smith-Waterman score is a classical choice (Smith and Waterman, 1981; Gusfield, 1997). Even if this algorithm relies on dynamic programming, thus making it very efficient (and accurate), its complexity is in $\mathcal{O}(n^2\ell^2)$ if n is the number of barcodes or reads, and ℓ their length. This becomes prohibitive for large datasets. A classical way to circumvent this difficulty is to use kmer-based distances (Sun et al., 2009; Mahé et al., 2014), a priori with a decrease in the quality of the estimation of the dissimilarity, but much faster to compute. A comparison between Smith-Waterman scores and kmer-based distances can be found in Sun et al. (2009). The question here is to know whether the loss in quality remains acceptable and does not lead to a decrease in adequacy between the botanical and the molecular-based classifications.

If the morphology-based taxonomic classification is a priori unique, this is not true for a molecular-based classification. A diversity of softwares for implementing hierarchical

clustering has been proposed for a decade or so in metabarcoding with the objective of efficient scaling with respect to the growing size of environmental datasets. This includes Uclust (Edgar, 2010), ESPRIT (Sun et al., 2009) and SWARM (Mahé et al., 2014) which make it possible to cluster millions of barcodes on a laptop. All of the Hierarchical Clustering algorithms mentioned above rely on heuristics (like computing kmer-based distances, considering short distances only i.e. working with sparse distance matrices) to make computation feasible within a reasonable time with reasonable memory. In this study, we focus on understanding the adequacy (or not) between molecular-based numerical classification and botanical classification, without computational constraints. We therefore consider Aggregative Hierarchical Clustering (AHC), whose above-mentioned algorithms can be seen as heuristic versions for scaling up, with three different aggregation methods (Single Linkage, Complete Linkage, Ward (Murtagh, 1983; Müllner, 2013)). Statistical models like Bayesian classifiers with mixture models have also been considered to cluster sequences (Hao et al., 2011). To extend the scope of statistical modeling in molecular-based taxonomy, we explore here the potential interest of a model-based clustering method, the Stochastic Block Model (SBM, Holland et al., 1983; Daudin et al., 2008; Lee and Wilkinson, 2019) as an alternative to AHC. SBM are already widely applied with success in domains like the social sciences (Barbillon et al., 2017), the analysis of ecological interaction networks (Miele and Matias, 2017) and neurology (Faskowitz et al., 2018). They rely on a more flexible definition of a cluster than AHC (searching for general groups and not just communities), and we hypothesized that SBM and AHC could be complementary in their capacity to distinguish meaningful groups of individuals in an inventory.

In the following section, we provide a brief description of the dataset. We also describe the method. Results on the quality of the adequacy between molecular-based and morphological-based classifications obtained on replicates are presented in Section 3.2, the results obtained on the whole dataset are presented in Section 3.3.

2 Materials and methods

2.1 Dataset and computation of dissimilarities

This study was conducted on a dataset built from a collection of some 1,500 trees located in the "Piste de Saint-Elie" experimental plot in French Guiana, mainly composed of lowland tropical rainforest (Sabatier et al., 1997). The data used here are part of a dataset gathered for the study published in Caron et al. (2019), which focused on adequacy or not between botanical-based and molecular-based classification at the species level over a wide range of diversity along the angiosperm tree. The principal result in Caron et al. (2019) is that molecular-based clustering is highly consistent with species delineation in a majority of cases, and that introgression or incomplete lineage sorting are the most likely explanations in the case of non-adequacy. We focus here on a similar question, at the level of genera, families and orders. The main elements for the material are recalled here, and the reader can refer to Caron et al. (2019) for details. Among this dataset, 1458 individual trees were selected for this study. For each tree, we used the botanical name as given by field botanists working with the Cayenne IRD Herbarium, and a sequence of chloroplastic marker trnH-psbA. These trees encompass 20 orders, 56 families, 182 genera and 428 species. Two 1458×1458 matrices of pairwise distances or dissimilarities between sequences were built, a first one using the Smith-Waterman algorithm for local sequence alignment (Smith and Waterman, 1981), and a second one for the distance between kmers distributions. The local alignment score is the most precise quantification of genetic dissimilarities between sequences, but it is time consuming. Several methods for building OTUs therefore rely on alternatives to local alignment scores. A classical way to circumvent this computational burden is to build kmer histograms for each sequence, and then compute the distance between the histograms. A kmer is a contiguous sub-sequence of length k in a given sequence. We selected short kmers here with $k = 4$ (there are $4^4 = 64$ different tetramers which is a good compromise between longer ones with more resolution, but too

sparse histograms, or smaller ones with coarse resolution and less empty categories). We designed an efficient algorithm that counts the frequency of each tetramer in each sequence, and a short program that computes a distance between any pair of frequency distributions as the ℓ^1 norm, i.e., the sum of absolute values of difference per tetramer.

The dataset used in the rest of this paper is composed of three files (see Frigerio et al. (2021)):

- a csv file of botanical names for each sequence for order, family, genus and species;
- a csv file of pairwise dissimilarities computed with the Smith-Waterman algorithm;
- a csv file of pairwise distances based on the comparison of tetramer histograms (same format as Smith-Waterman dissimilarities)

2.2 Visualisation of the whole dataset using MDS

A preliminary step is to propose a global picture of the dataset based on the dissimilarity matrices, without a classification objective. Multidimensional Scaling (MDS) is a method that, once a dissimilarity matrix is given, builds a point cloud in a Euclidean space of prescribed dimension where each point corresponds to a sequence, and such that the Euclidean distance between any two points is as close as possible to the dissimilarity given in the matrix (Cox and Cox, 2001; Izenman, 2008). In our case, we selected the so-called Classical Scaling, as proposed initially in Torgerson (1952). It is expected that the projection of the point cloud on the first axis encompasses much of the information about the structure of the point cloud. MDS was run with the dissimilarity matrices built with the Smith-Waterman algorithm and as distances between kmer histograms. To perform MDS, we have filtered the whole data set by keeping sequences in orders with 15 specimen or more only.

2.3 General approach

Depending on the specific question addressed, we did not work with the same sample of the whole dataset. However, in all cases, the general approach for comparing two classifications was the same and can be broken down into four steps.

In step 1, we selected the sub-sample: either the whole dataset with filters, or only the individuals of a particular order, or a particular family. We then extracted the sub-matrix corresponding to the n individuals in the sample from the global dissimilarity matrix.

In step 2, we built the classifications corresponding to AHC with the three aggregation methods, Ward, Complete Linkage (CL) and Single Linkage (SL), and to SBM (see SI for a description of these methods). The number of clusters K was provided by the botanical classification of the individuals of the sub-sample. For instance if we wanted to study adequacy between the classification into families and the molecular-based ones, we cut the AHC hierarchy of classifications at K equal to the number of families in the sample, and we ran SBM for the same value. At the end of step 2, we had five different classifications of the individuals in the sub-sample.

In step 3, we compared the classifications, two by two, for each possible pair of classifications (10 pairs in total). We used a visual tool for preliminary analysis of the adequacy between two classifications, with Sankey plots. A Sankey plot is a flow chart in which the width of an arrow is proportional to the flow. For instance, if there are $n_{kk'}$ sequences that are in class k for the botanical classification and k' for a numerical classification, there is a flow of intensity $n_{kk'}$ between those two clusters. We computed an index as well, to quantify the adequacy. Classification comparison is equivalent to the comparison of two partitions of the same set, a dynamic research area with several surveys (Pfitzner et al., 2009). Several indices were proposed and we chose the Normalised Mutual Information (NMI1 in Pfitzner et al., 2009), which we will refer to as NMI (see SI for a formal definition). It is not empirical and has a sound basis in information theory, as opposed to

indices based on counting pairs of elements that may be non-symmetric or non-bounded or even be dependent on K or n, making comparison difficult. NMI is normalised and, as such, bounded by 0 and 1, facilitating interpretation and comparison of indices. A NMI of 0 indicates independence between the two classifications, while a NMI of 1 indicates a perfect adequacy.

Finally, when replicates of the experiment are performed like in Section 2.4, in a fourth step, we analysed the distributions of the NMIs for a given pair of classifications in order to study trends in the adequacy, using histogram representation, boxplots and computing statistics on the distribution (mean, median, etc).

2.4 Comparison of botanical and molecular-based classification at the family and genus levels, on replicates

We worked with 30 sub-samples of the whole dataset, corresponding to the individuals of 10 orders and 20 families. We selected only orders (resp. families) composed of at least 15 individuals, and structured into more than one family (resp. genus). We then looked for the families present in one order or the genera present in one family. The orders are Malpighiales, Ericales, Sapindales, Laurales, Myrtales, Magnoliales, Gentianales, Rosales, Oxalidales and Malvales.

We structured the empirical analysis of the NMIs obtained (30×10 values) into four different analyses addressing the following questions: (i) What is the level of adequacy between the botanical classification and each of the four molecular-based ones? (ii) Are the classifications provided by the four molecular-based clusterings similar? (iii) Can we identify elements of the dissimilarity matrix that explain the variability observed in the answer to question (i) and that would be indicators of the adequacy/independence between the botanical classification and the molecular-based ones? (iv) How does the adequacy change between the botanical classification and the molecular-based ones when substituting

kmer-based distances for SW dissimilarities? In practice, for question (i), we considered only the NMIs involving the botanical classification and any of the four molecular-based ones, whereas for question (ii), we only considered the NMIs between any pairs of the molecular-based classifications. For question (iii), we studied three factors: the taxonomic level of the groups, the entropy of the botanical classification (defined as the entropy of the normalised vector of the group size), and the structure of the dissimilarity matrix. The latter was measured by three different ratios between the within-group dissimilarities and the between-group dissimilarities (see SI). We only present the one here that lead to the most interesting results, r_{mean}, defined as the mean of the larger within-class dissimilarity over the mean of the smaller between-class dissimilarity.

2.5 Comparison of botanical and molecular-based classification with a wide scope of diversity

We looked at whether or not clustering on the whole dataset could directly retrieve botanical classification at levels higher than species (genera, families, orders). In addition, the same comparison was done for species as well, as a benchmark. Since several taxa are singletons, regardless of the level, or have a very small number of sequences (e.g. Apiales are represented by three sequences only in the whole sample), we built one subsample for each taxonomic level by filtering out taxa with less sequences than a given threshold. The size of those subsamples are given in Table 1, with the number of sequences and of different taxa per level, and the threshold selected for filtering sequences.

We ran SBM and Aggregative Hierarchical Clustering (AHC) with Ward, CL and SL, both on the matrix of dissimilarities between scores of the Smith-Waterman algorithm and on distances between tetramer frequencies. We compared each of these four classifications with the botanical one using NMI. As for the study of the replicates, we also computed the entropy and the r_{mean} ratio of the botanical orders, families, genera and species clas-

sifications. For each of the taxonomic levels, we produced a visual graphical analysis by generating Sankey plots.

3 Results

3.1 Visualisation of the whole dataset using MDS

We represented the shape of the point cloud on the first two axes built with MDS on the dissimilarity matrix, with points coloured according to the order that they belong to (see Figures 1 and 2). For Smith-Waterman-based dissimilarities, axis 1 clearly distinguishes Ericales (in purple) and Sapindales (dark green), and axis 2, Malpighiales (in light green). Axis 3 distinguishes Fabales (blue), and the set of Laurales and Magnoliales (red and orange), which are primitive Eudicots. The organisation of the point cloud is different for kmer-based dissimilarities. The point cloud is more compact. Axis 1 distinguishes the same set of Laurales and Magnoliales, and axis 2 distinguishes Fabales. Clearly, the shape of the point cloud based on Smith-Waterman distances is more closely related to the organisation of specimens in botanical orders. Such a connection is blurred for kmer-based distances. This allowed us to predict that adequacy between the botanical classification and the molecular-based ones will be lower when using kmer-based distances.

3.2 Comparison of botanical and molecular-based classification at the family and genus levels on replicates

We present first the results obtained with SW scores for points (i) to (iii) raised in Section 2.4. We then show how results change when working with kmer-based distances (point (iv)).

(i) **Level of adequacy between the botanical classification and the molecular-based ones.** For SBM, Ward, SL and CL, the shape of the histogram of the 30 NMIs

is the same (see Figure 1 of SI). The mode is observed at large values: NMIs are larger than 0.9 for one-third of the replicates. Then, intermediate values of the NMI are not often observed, in particular, in the case of the NMIs between SL and the botanical classification. Median values of the histograms vary between 0.68 (for SBM) and 0.87 (for Ward) (see Table 3). So globally we observe a good to very good adequacy between the botanical and the molecular-based classification, with better performance for the Ward method.

(*ii*) **Mutual adequacy of the responses of the four molecular-based clustering methods.** There is a strong adequacy between the three AHC methods. We observed larger NMIs between Ward and CL than between Ward and SL or CL and SL. The SBM classification is globally in good adequacy with the three AHC ones (NMI median larger than 0.64) and is closer to the CL and the Ward classifications (see Table 4 with statistics of the NMI distributions).

(*iii*) **Factors explaining variability in the results.** When pooling the results for the four molecular-based classifications, we observed no clear difference in the distribution of the NMIs (NMIs are between the botanical classification and the molecular-based ones) computed on replicates whose groups are at the family level and those at the genus level. We observed a trend towards an increase in adequacy between botanical and molecular-based classifications when the entropy of the sub-sample increases (Figure 3 left side). This effect is observed for both dissimilarities.

We also observed a clear decrease of the adequacy when r_{mean} increases (see Figure 3 right side). Adequacy is good to strong (larger than 0.7) for a ratio lower than 1. When a dissimilarity matrix is associated with a large ratio, it means that several sequences exist that are closer to sequences belonging to a different genus or family than to sequences in their own genus or family.

(*iv*) **Influence of the choice of dissimilarity.** We observed a systematic decrease of the NMIs when substituting the SW dissimilarity with the kmer-based distance (Table 2). This decrease ranged between 5 % to 39 % depending on the taxonomic level of the

groups and the molecular-based clustering method, and it appear to be higher for SL. The adequacy with the botanical classification is the highest for the Ward-based classification, and we still observed the influence of the entropy of the botanical classification and of r_{mean} on the adequacy (Figure 2 in SI).

In conclusion, adequacy between the botanical classification and molecular-based ones can be good to very good. However, there are also situations were the adequacy is low. We have identified several factors that can influence the level adequacy: the choice of the clustering method, with Ward leading to the greatest adequacy; the choice of the dissimilarity, with a greater adequacy for SW dissimilarities than for kmer-based distances; the entropy of the botanical classification, with greater adequacy for larger entropies; r_{mean}, with greater adequacy for lower ratios.

3.3 Comparison of botanical and molecular-based classification with a wide scope of diversity

The results presented here extend the results on the points (i), (iii) and (iv) with four new experiments: we compared, on the one hand, the botanical classifications of the whole dataset partitioned into 11 orders, 20 families and 36 genera, as well as 55 species as a benchmark, (see Table 1) and, on the other, the molecular-based classifications obtained for the same number of classes.

(i) **Level of adequacy between the botanical classification and the molecular-based ones.** On Figure 4 one curve per method displays the evolution of the NMI along a gradient of taxonomic levels running over species, genera, families and orders. All curves, regardless of the molecular-based clustering method and the dissimilarity, display a decrease from species to orders. All of the methods are excellent for identifying species (NMI $\simeq$ 0.8), and decreases depend on the method: a slight decrease for the Ward method, a sharp

decrease for the SL method, and an intermediate decrease for CL or SBM. SL seems to display poorer indices regardless of the distance for orders or even families. To illustrate this, the correspondence between botanical, Ward and SBM classifications obtained with SW dissimilarities are graphically visualised in Figure 5 for orders and Figure 6 for families, with Sankey plots. We can note two types of behaviour: a botanical group is split into several groups in Ward or SBM classifications or, on the contrary a Ward or SBM group is composed of individuals from several botanical groups. The latter is more problematic when interpreting molecular-based classifications. On Figure 7, we can observe that the low NMI for SL at the order level is due to the creation of a giant cluster formed by almost all of the orders present in the dataset.

(*iii*) **Factors explaining variability in the results.** The fact that adequacy between the molecular-based and the botanical-based classifications decreases when the taxonomic level of the groups searched increases is in agreement with the influences of the entropy and of the r_{mean} observed on the replicates. Indeed entropy here decreases and r_{mean} increases when moving from the classification into species and genera towards families and orders (see Table 5).

(*iv*) **Influence of the choice of dissimilarity.** Regardless of the AHC method and the taxonomic level, NMI is higher for SW-based dissimilarities than with kmer-based distances. The decrease in the NMIs is stronger when the groups are families or orders: NMIs varied between 7% and 67% with the larger decrease observed for SL. SBM seems to be less sensitive to the measure of dissimilarities between sequences, with even, in some cases, a larger NMI with the kmer-based distances.

4 Discussion

In this study, several numerical methods were compared on a well-known dataset of approximately 1,500 specimens of trees in a French Guianese forest for the purpose of quan-

tifying the adequacy between, on the one hand, botanical classification and, on the other, molecular-based classification on an array of genetic distances, on deep nodes of classifications. We discuss here the results provided for this purpose.

4.1 Adequacy between botanical and molecular-based classifications

There is one pattern common to the study based on the clusterings of the 30 replicates and the clusterings performed on the whole dataset: regardless of the combination between taxonomic level and dissimilarity, AHC with the Ward aggregation criterion provides the best adequacy. Other methods rank differently depending on these combinations. Adequacy can be high, in particular when the molecular-based clustering is based on the SW dissimilarity. However, we also occasionally observed low adequacy, and we will discuss the reasons for this. When interpreting NMI values, it is important to have in mind that NMI is conservative in the sense that a strong adequacy is required to obtain a large NMI value. The strength of the adequacy could be higher with another choice of index, but we selected NMI partly for this conservative behaviour.

A strong assumption in our study is that the number of groups K in the botanical classification is known when performing the molecular-based clustering. This is obviously not the case in real situations, like in metabarcoding of environmental samples. When K is not available, classical tools for model selection can be used to estimate a number of groups that lead to a trade-off between a good explanation of the dissimilarity matrix and parsimony. However, these criteria do not include a goal of adequacy with the botanical classification. In White et al. (2010), to compare molecular-based clustering at the OTU level and the taxonomic classification, the authors used partial assignment of the sequences and the VI-cut method to automatically determine the number of OTUs. The method relies on the Value of Information to compare two clusterings, which we did not select for our

study because it is not normalised. However, the VI-cut method could easily be extended to the NMI index and therefore provide a way to estimate the number of groups, driven by the partial taxonomic knowledge that is available on some sequences of the inventory.

Although neighbor-joining (Saitou and Nei, 1987) is one of the reference methods in phylogenies, and based on distances, we have not retained it in our study for two reasons. First, the adequacy between orders and clades[2] (monophyly of orders) in the tree is not excellent (see section 5 and Figure 5 in SI), and second, neighbor-joining is not a clustering method: the outcome cannot be automatically organized as a partition into clusters.

4.2 Adequacy of botanical classification and the AHC classifications

In our result, a variability of quality is observed according to the linkage method. If the dataset is organised as a set of isolated clusters, all linkage methods will find it and provide the same classification. If not, different linkage methods will yield different classifications. Not surprisingly, we recover these behaviours in our experiments on molecular-based clustering of the tree specimens.

Ward method: The Ward method nearly always led to the best adequacy with botanical classification, regardless of the measurement of distance (SW or kmers) and the taxonomic level of the groups (Sections 3.2 and 3.3).

Complete linkage Method The CL method generally led to the second-best adequacy with the botanical classification. It provided classifications very similar to those obtained with the Ward method (see Table 4).

[2]A clade here is an internal node with its descent.

Single linkage method: On the contrary, adequacy between the classification provided by the SL method and the botanical classification was highly variable and could be either very good or very poor. The adequacy was very poor with the classification into orders of the whole dataset (NMI = 0.06 for SW dissimilarities and NMI = 0.02 for kmer-based distances, which is very close to independence), better but still low for the classification into families (NMI = 0.44 for SW dissimilarities, and NMI = 0.34 for kmer-based distances). A reason for that can be seen on Figure 7: the SL classification is composed of a huge cluster, containing sequences from all orders, and a set of much smaller clusters, each containing one, seldom two, orders. The creation of the huge cluster may be due to low dissimilarities existing between the orders. By nature, the SL criterion will link these orders. This is a well-known drawback of SL.

4.3 Interest of SBM models for molecular-based classification

Even if the SBM clustering and the botanical one are in very good adequacy in some of the experiments, globally, the NMI values for SBM are lower than the NMIs for the best AHC method (see Table 2 and Figure 4). When adequacy with the botanical classification is good, then the SBM classification resembles the one obtained with the Ward method (Table 4). This is the case where the dissimilarity matrix is well structured into communities, and all clustering methods will perform well. When adequacy is low, our interpretation is the following. The main difference between AHC and SBM clustering is that AHC looks for groups with small within-group dissimilarity (communities), while SBM seeks for groups such that an individual in group k shares, with the other members of group k, the same pattern of connections with the other groups, and members of group k are almost at the same distance to each others. However, this distance is not necessarily small, meaning that SBM groups should not be systematically interpreted as communities. When the matrix of the pairwise dissimilarities is not clearly structured according to the botanical groups, SBM clustering can create groups with individuals that are far from each other. This is

what we observed on the SBM classification of the whole dataset into orders (both for the SW and the kmer-based clustering). For several SBM groups, the estimated parameter characterising the mean within group distance (mean of the Poisson distribution or of the Gaussian distribution) was larger than the lower mean distance with the other groups. In these situations, the NMI between the botanical and the SBM classification will obviously be low, and the ratio r_{mean} will be large. A SBM classification with groups of large within-group mean distances should be a warning that the matrix of dissimilarities is not entirely structured according to the botanical classes. For similar reasons, SBM will also be able to detect outlier individuals by gathering them into a group, while methods looking for communities will force them to enter a community. For these reasons, we think SBM should be considered as a valuable tool for (meta)barcoding.

4.4 Factors explaining the variations in the adequacy.

One of the two main factors influencing the quality of adequacy between the botanical and the molecular-based classifications is the relative difference between the dissimilarities within and between groups in botanical classification. This notion was well captured by the r_{mean} ratio and, we obtained a clear tendency for NMIs to decrease when the ratio increases on the 30 replicates (Figure 3). When considering the clustering of the whole dataset, the same tendency was observed. The other factor influencing the quality of adequacy is the value of the entropy of the distribution of the group sizes in the botanical classification. We observed a tendency for an increase in adequacy when entropy increases, both on the 30 replicates and when clustering the whole dataset at different taxonomic levels.

In the latter experiments we obtained a clear decrease of adequacy when the taxonomic level increases, whereas in the experiments on the 30 replicates, adequacy was better at the family level than at the genus level. These apparent contradictory results are actually explained by the fact that they correspond to two different protocols. On the 30 replicates the targeted set of sequences to cluster is different for each replicate: we did not search for

families and genera among the same set of individuals. We instead searched for families (resp. genera) of sequences of the same order (resp. family).

The negative influence of the r_{mean} ratio and the positive influence of the entropy are global tendencies. We also observed variations around these main tendencies, which means that they are probably not the only factors explaining the NMI values. Still, they are strong signals.

4.5 Biological interpretation

We can expect that it is more difficult to cluster a set of individuals in a right way when the number of targeted groups increases (more possibilities to ill-classify). It is not that simple and actually depends on the structure of the dissimilarity matrix. Indeed, in our study on the whole dataset, the adequacy between the molecular-based and the botanical-based classification is better when groups are at a low taxonomic level, hence more numerous, regardless of the method and the distance (see Figure 4). As discussed in Section 4.4, the r_{mean} ratio, implying distances within a group over distances between groups, is smaller when groups are families than when they are orders. This suggests that families are better delineated than orders by pairwise distances. The results shown in Figure 4 extend this observation to species over genera, and show that molecular-based delineation of taxa is more accurate at fine taxonomic levels than at coarse ones. This can be related to the observation that only species have a sound biological meaning, whereas genera, families and orders are constructions based on similarities.

4.6 Comparison between SW and kmer-based dissimilarities

Computing SW dissimilarity between two sequences is the most precise way to compare them. However, it is time-consuming. Computing kmer-based distances is much quicker, but at the cost of approximations. The two 1D histograms of SW dissimilarities and kmer-

based distances are provided on Figures 3 and 4 of SI). A coarse correlation can be observed between both quantifications of dissimilarities (see Figure 8), stronger for small dissimilarities. However, a significant number of pairs of sequences exists with a very low SW dissimilarity and a significant kmer-based distance. This is due to the high variability in length of the trnH marker. For instance, a small SW dissimilarity means that the smallest sequence is nearly identical to a contiguous sub-sequence of the larger one. However, due to the dissimilarity in length of the two sequences, the kmer histograms cannot be similar, and the kmer-based distance is large. Therefore, some small values of the SW dissimilarity can be associated with median values of the kmer-based distance. Since the AHC classification (regardless of the linkage) builds groups of individuals with small within group distances, it can be expected that the SW-based and the kmer-based classifications will be different. In practice, as expected, we observed that adequacy decreases when substituting kmers for SW regardless of the combination between the taxonomic level and the clustering method. However, substituting kmer-based distances for SW dissimilarities did not lead to a real collapse of molecular-based methods.

It can be observed that NMIs between the botanical classification and the SBM one seem to be less sensitive to the dissimilarity used than NMIs between botanical and AHC classifications (see Figure 4 and Table 2). This is probably due to the fact that, as opposed to AHC, SBM does not build the groups on a criterion that minimises within-group distances, as explained in section 4.3.

4.7 Perspectives for metabarcoding

The dataset is sufficiently small for all calculations to be run on a laptop in a reasonable time, making it possible to focus on a comparison of the methods. Some methods are clearly more accurate than others to retrieve orders or families in our dataset. The expectation is that those methods are those that will permit inventories or clustering at higher taxonomic levels like families, orders or phyla in metabarcoding. We therefore make two

recommendations.

First, we recommend using AHC with the Ward method for clustering regardless of the taxonomic level, and not using AHC with Single Linkage which may produce poor results, despite the observation that current softwares scaling up with NGS massive datasets make it possible to use it (like MOTHUR) or yield results very close to it (like SWARM, see Mahé et al., 2014). It can be observed that SWARM, close to SL, yields giant clusters that are cut into pieces by a sort of pruning.

Second we recommend using SBM classification to detect, via the estimated distribution of within cluster distances, situations where the molecular-based classifications may be poorly related to the morphological-based one (because the dissimilarity matrix is not clearly structured into communities).

These results and observations lead us to recommend the pursuite of methodological efforts to analyse metabarcoding data for building inventories at the coarse level (i.e., between phyla and orders). Inventories at the coarse level are a first step towards the global exploration of diversity of unknown groups. This can be done in two ways. First, nearly all surveys about clustering emphasize that there is no method that is perfect and better than some others for all evaluation criteria (see, e.g., Fahad et al., 2014). Therefore, it may be useful to produce classifications by several numerical methods and to extract the shared elements. These are the ones in which the user can be more confident that they effectively reflect an actual structure in the data. Second, scaling-up methods that have proven themselves to properly perform on well-known datasets, like AHC with Ward linkage or SBM-based clustering, is a key issue. A very efficient method for clustering may be like "divide and conquer": first, dividing the problem by building connected components in a graph built from pairwise distances and, second, conquering by implementing AHC Ward or SBM in each connected component.

More globally, connecting these efforts with studies on wider classes of methods under development for clustering for big-data (Fahad et al., 2014) is a challenging issue for

metabarcoding.

5 Acknowledgements

The authors acknowledge support of an “Investissement d’Avenir” grant managed by the Agence Nationale de la Recherche (CEBA, ref. ANR-10-LABX-25-01). M.A. Abouabdallah PhD is funded by INRAE and by INRIA. The authors acknowledge the work done by H. Caron (sequencing), D. Sabatier and J.F. Molino (botanical assignation of trees) for a previous work that we have used here. We thank Jean-Marc Frigerio for many useful discussions for data management and analysis, as well as Philippe Chaumeil for his contribution to the programs to compute dissimilarities.

6 Author contributions

A.F. and N.P. designed the study. M.A.A., A.F. and N.P. performed the research and analyzed the data. The paper was drafted by M.A.A., A.F. and N.P. and written by A.F. and N.P. All authors commented on and approved the final manuscript.

7 Data accessibility

Sequences used to compute distances are in NCBI under accession number KX247940–KX249593. A fasta file with these sequences, a file with taxonomical assignation for each tree, as well as pairwise Smith-Waterman and kmer distances are publicly available at https://doi.org/10.15454/XSJ079 in Inrae Data Portal.

References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410.

Barbillon, P., Donnet, S., Lazega, E., and Bar-Hen, A. (2017). Stochastic block-models for multiplex networks : An application to a multilevel network of researchers. *Journal of the Royal Statistical Society: Series A Statistics in Society*, 180(1):295–314.

Bik, H. M., Porazinska, D. L., Creer, S., Caporaso, J. G., Knight, R., and Thomas, W. K. (2012). Sequencing our way towards understanding global eukaryotic biodiversity. *Trends in Ecology and Evolution,*, 27:233–243.

Blaxter, M., Mann, J., Chapman, T., Thomas, F., Whitton, C., Floyd, R., and Abebe, E. (2005). Defining operational taxonomic units using DNA barcode data. *Philosophical Transactions of the Royal Society B*, 360:1935–1943.

Caron, H., Molino, J.-F., Sabatier, D., Léger, P., Chaumeil, P., Scotti-Saintagne, C., Frigério, J.-M., Scotti, I., Franc, A., and Petit, R. J. (2019). Chloroplast DNA variation in a hyperdiverse tropical tree community. *Ecology and Evolution*, 9(8):4897–4905.

Cole, A. J., editor (1969). *Numerical Taxonomy.* Academic Press, London & New York.

Cox, T. and Cox, M. A. A. (2001). *Multidimensional Scaling - Second edition*, volume 88 of *Monographs on Statistics and Applied Probability.* Chapman & al.

Daudin, J.-J., Picard, F., and Robin, S. (2008). A mixture model for random graphs. *Statistics and Computing*, 18:173–183.

Edgar, R. C. (2010). Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, 26:2460–2461.

Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A. Y., Foufou, S., and Bouras, B. (2014). A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis. *IEEE Transactions on Emerging Topics on Computing*, 2(4):267–279.

Faskowitz, J., Yan, X., Zuo, X.-N., and Sporns, O. (2018). Weighted stochastic block models of the human connectome across the life span. *Scientific Reports*, 8.

Felsenstein, J. (2004). *Inferring phylogenies.* Sinauer.

Floyd, R., Abebe, E., Papert, A., and Blaxter, M. (2002). Molecular barcodes for soil nematode identification. *Molecular Ecology*, 11:839–850.

Fontaneto, D., Boschetti, C., and Ricci, C. (2008). Cryptic diversification in ancient asexuals: evidence from the bdelloid rotifer *Philodina flaviceps*. *Journal of Evolutionary Biology*, 21:580–587.

Frigerio, J.-M., Caron, H., Sabatier, D., Molino, J.-F., and Franc, A. (2021). Guiana Trees. Portail Data INRAE, V1, DOI: 10.15454/XSJ079.

Gusfield, D. (1997). *Algorithms on strings, trees, and sequences.* Cambridge University Press, Cambridge, UK.

Hajibabaei, M., Shokralla, S., Zhou, X., Singer, G. A. C., and Baird, D. J. (2011). Environmental barcoding: a next generation sequencing approach for biomonitoring applications using river benthos. *PLoS One*, 6(4):e17497.

Hao, X., Jiang, R., and Chen, T. (2011). Clustering 16S rRNA for OTU prediction: a method of unsupervised Bayesian clustering. *Bioinformatics*, 27(5):611–618.

Hebert, P. D. N., Cywinska, A., Ball, S. L., and deWaard, J. R. (2003). Biological identifications through DNA barcodes. *Proceedings of the Royal Society of London B*, 270:313–321.

Hillis, D. M., Moritz, C., and Mable, B. (1996). *Molecular Systematics*. Sinauer, Sunderland, Mass.

Holland, P., Laskey, K., and Leinhardt, S. (1983). Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137.

Izenman, A. J. (2008). *Modern Multivariate Statistical Techniques*. Springer, NY.

Ji, Y., Ashton, L., Pedley, S. M., Edwards, D. P., Tang, Y., Nakamura, A., Kitching, R., Dolman, P. M., Woodcock, P., Edwards, F. A., Larsen, T. H., Hsu, W. X., Benedick, S., Hamer, K. C., Wilcove, D. S., Bruce, C., Wang, X., Levi, T., Lott, M., Emerson, B. C., and Yu, D. W. (2013). Reliable, verifiable and efficient monitoring of biodiversity via metabarcoding. *Ecology Letters*, 16(10):1245–1257.

Kermarrec, L., Franc, A., Rimet, F., Chaumeil, P., Humbert, J.-F., and Bouchez, A. (2013). Next-generation sequencing to inventory taxonomic diversity in eukaryotic communities: a test for freshwater diatoms. *Molecular Ecology Resources*, 13:607–619.

Lee, C. and Wilkinson, D. (2019). A review of stochastic block models and extensions for graph clustering. *Applied Network Science*, 4(122).

Leray, M. and Knowlton, N. (2015). DNA barcoding and metabarcoding of standardized samples reveal patterns of marine benthic diversity. *PNAS*, 112(7):2076–2081.

López-García, P., Rodriguez-Valera, F., Pedros-Alio, C., and Moreira, D. (2001). Unexpected diversity of small eukaryotes in deep-sea Antarctic plankton. *Nature*, 409:603–607.

Mahé, F., Rognes, T., Quince, C., de Vargas, C., and Dunthorn, M. (2014). Swarm: robust and fast clustering method for amplicon-based studies. *PeerJ*, 2:e593.

Miele, V. and Matias, C. (2017). Revealing the hidden structure of dynamic ecological networks. *Royal Society Open Science*, 4(6).

Munch, K., Boomsma, W., Huelsenbeck, J. P., Willerslev, E., and Nielsen, R. (2008). Statistical Assignment of DNA Sequences Using Bayesian Phylogenetics. *Systematic Biology*, 57(5):750–757.

Murtagh, F. (1983). A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359.

Müllner, D. (2013). fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python. *Journal of Statistical Software*, 53(9):1–18.

Pfitzner, D., Leibbrandt, R., and Powers, D. (2009). Characterization and evaluation of similarity measures for pairs of clusterings. *Knowledge and Information Systems*, 19(361).

Pons, J., Barraclough, T. G., Gomez-Zurita, J., Cardoso, A., Duran, D. P., Hazell, S., Kamoun, S., Sumlin, W. D., and Vogler, A. (2006). Sequence-Based Species Delimitation for the DNA Taxonomy of Undescribed Insects. *Systematic Biology*, 55(4):595–609.

Puillandre, N., Modica, M. V., Zhang, Y., Sirovich, L., Boisselier, M.-C., Cruaud, C., Holford, M., and Samadi, S. (2012). Large-scale species delimitation method for hyperdiverse groups. *Molecular Ecology*, 21:2671–2691.

Sabatier, D., Grimaldi, M., Prévost, M., Guillaume, J., Godron, M., Dosso, M., and Curmi, P. (1997). The influence of soil cover organization on the floristic and structural heterogeneity of a guianan rain forest. *Plant Ecology*, 131:81–108.

Saitou, N. and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, (4):406–425.

Smith, P. D. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197.

Sneath, R. H. A. and Sokal, R. R. (1973). *Numerical taxonomy*. Freeman, San Francisco.

Sogin, M. L., Morrison, H. G., Huber, J. A., Welch, D. M., Huse, S. M., Neal, P. R., Arrieta, J. M., and Herndl, G. J. (2006). Microbial diversity in the deep sea and the underexplored "rare biosphere". *PNAS*, 103(32):12115–12120.

Sun, Y., Cai, Y., Liu, L., Yu, F., Farrell, M. L., McKendree, W., and Farmerie, W. (2009). ESPRIT: estimating species richness using large collections of 16S rRNA pyrosequences. *Nucleic Acids Research*, 37(10):e76.

Taberlet, P., Coissac, E., Pompanon, F., Brochman, C., and Willerslev, E. (2012). Towards next-generation biodiversity assessment using DNA metabarcoding. *Molecular Ecology*, 21:2045–2050.

Talavera, G., Dinca, V., and Vila, R. (2013). Factors affecting species delimitations with the GMYC model: insights from a butterfly survey. *Methods in Ecology and Evolution*, 4:1101–1110.

The Angiosperm Phylogeny Group, Chase, M. W., Christenhusz, M. J. M., Fay, M. F., Byng, J. W., Judd, W. S., Soltis, D. E., Mabberley, D. J., Sennikov, A. N., Soltis, P. S., and Stevens, P. F. (2016). An update of the Angiosperm Phylogeny Group classification for the orders and families of flowering plants: APG IV. *Botanical Journal of the Linnean Society*, 181(1):1–20.

Torgerson, W. S. (1952). Multidimensional Scaling: I. Theory and Method. *Psychometrika*, 17(4):401–419.

White, J. R., Navlakha, S., Nagarajan, N., Ghodsi, M.-R., Kingsford, C., and Pop, M. (2010). Alignment and clustering of phylogenetic markers - implications for microbial diversity studies. *BMC Bioinformatics*, 11(152):1471–2105.

Yang, Z. (2006). *Computational Molecular Evolution*. Oxford Series in Ecology and Evolution. Oxford University Press.

Zhang, J., Kapli, P., Pavlidis, P., and Satamakis, A. (2013). A general species delimitation method with applications to phylogenetic placements. *Bioinformatics*, 29(22):2869–2876.

8 Tables

Taxonomic level	Number of sequences	Number of taxa	Minimal size of a taxon
Species	313	55	5
Genera	845	36	10
Families	1349	30	10
Orders	1357	11	15

Table 1: Properties of the four subsamples of sequences, one per taxonomic level. The number of sequences in the sample is lower for low taxonomic levels because we selected only taxa composed of a minimal number of sequences, and there are less sequences of a given species than of a given genera, etc. Each subsample is used for a comparison between the molecular-based clustering methods and the botanical classification.

		Families		Genera		Pooled	
Method		SW	kmers	SW	kmer	SW	kmer
AHC	Ward	1	0.61	0.83	0.73	0.87	0.71
	SL	0.88	0.54	0.75	0.59	0.76	0.58
	CL	0.85	0.63	0.75	0.71	0.75	0.67
SBM		0.57	0.52	0.82	0.66	0.68	0.63

Table 2: Normalised Mutual Information between the botanical classification (into families or into genera) and the four molecular-based classifications (row) for two dissimilarities (column). SW stands for Smith-Waterman and kmer for kmer-based distances computed with kmers of length $k = 4$. Results for families are median values over 10 samples and results for genera are median values over 20 samples. A sample is the set of sequences of an order (10 of them) or a family (20 of them).

Pairs of methods	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
SL vs. botanical	0.013	0.27	0.76	0.64079	1.00	1.00
Ward vs. botanical	0.05	0.40	0.87	0.73	1.00	1.00
CL vs. botanical	0.03	0.47	0.75	0.68	1.00	1.00
SBM vs. botanical	0.10	0.33	0.68	0.64	0.98	1.00

Table 3: Statistics on the distribution of the Normalised Mutual Information (NMI) computed between each molecular-based classification and the botanical one. The statistics are computed on the NMIs obtained on 30 replicates (10 classifications into families and 20 into genera). A sample is the set of sequences of an order (10 of them) or a family (20 of them). Results obtained using the Smith-Waterman dissimilarity.

Pairs of methods	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
SBM Vs SL	0.039	0.30	0.65	0.59	0.98	1.00
SBM vs. Ward	0.31	0.52	0.74	0.74	1.00	1.00
SBM vs. CL	0.11	0.47	0.65	0.67	0.91	1.00
SL vs. Ward	0.04	0.34	0.78	0.69	1.00	1.00
SL vs. CL	0.05	0.56	0.84	0.74	1.00	1.00
Ward vs CL	0.05	0.70	0.86	0.80	1.00	1.00

Table 4: Statistics on the distribution of the Normalised Mutual Information(NMI) computed between each pair of molecular-based classifications. The statistics are computed on the NMIs obtained on 30 replicates (10 classifications into families and 20 into genera). A sample is the set of sequences of an order (10 of them) or a family (20 of them). Results obtained using the Smith-Waterman dissimilarity.

	Orders	Families	Genus	Species
Entropy	2.15	3.01	3.39	3.98
r_{mean} with SW	2.22	1.3	0.60	0.30
r_{mean} with kmer	1.89	1.29	0.77	0.14

Table 5: Entropy and r_{mean} ratio (describing the ratio between mean within-group and mean between-group dissimilarities) for the botanical classifications of the dataset into orders, families, genera and species. SW stands for Smith-Waterman and kmer for kmer-based distances computed with kmers of length $k = 4$. Samples (one per taxonomic level) are those which have been built with the filters presented in table 1.

9 Figures

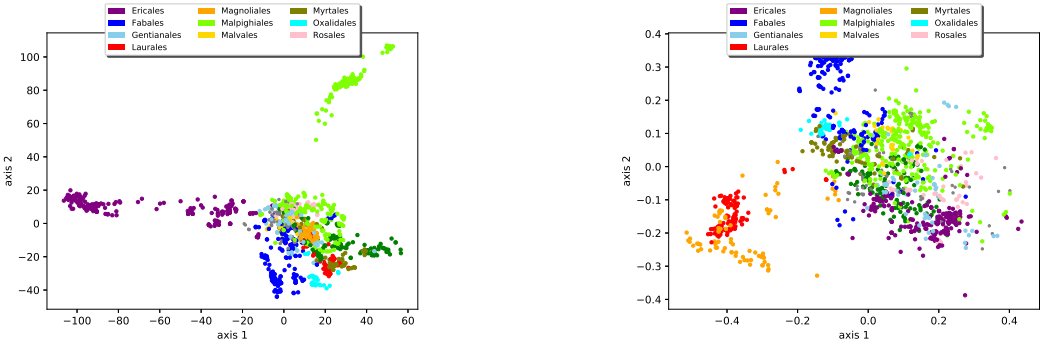


Figure 1: MDS on the dissimilarity matrix of the dataset of orders with 15 individuals or more, projected on axis 1 and 2. One dot is one sequence. Dissimilarities are computed with the Smith-Waterman algorithm on the left panel, and as distances between kmer histograms (length $k = 4$) on the right one.

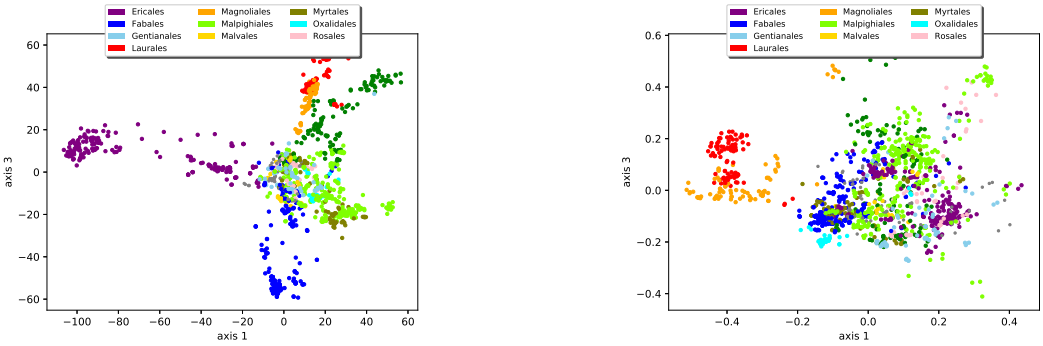


Figure 2: MDS on the dissimilarity matrix of the dataset of orders with 15 individuals or more, projected on axis 1 and 3. One dot is one sequence. Dissimilarities are computed with the Smith-Waterman algorithm on the left panel, and as distances between kmer histograms (length $k = 4$) on the right one.

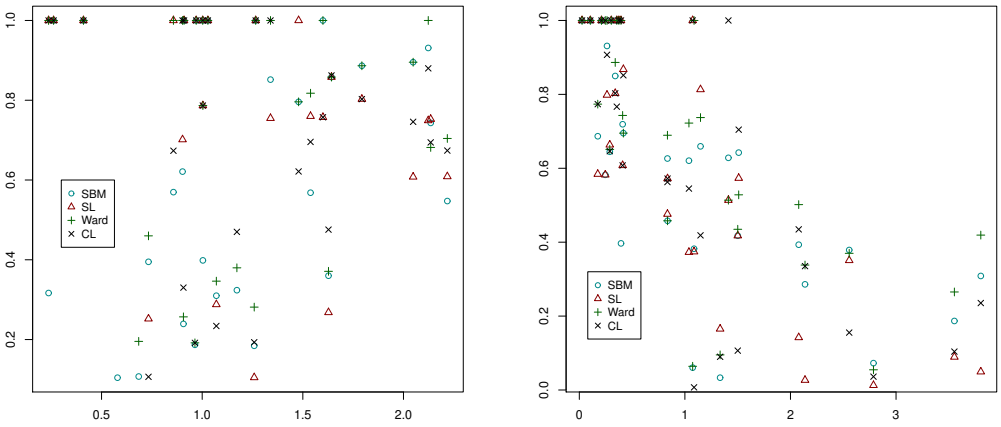


Figure 3: Normalised Mutual Information as a function of the entropy (left) and the ratio r_{mean} (right) computed on the botanical classification. Each point corresponds to one of the four molecular-based clustering method applied to one of the 30 replicates. Clustering is made using the Smith-Waterman dissimilarity.

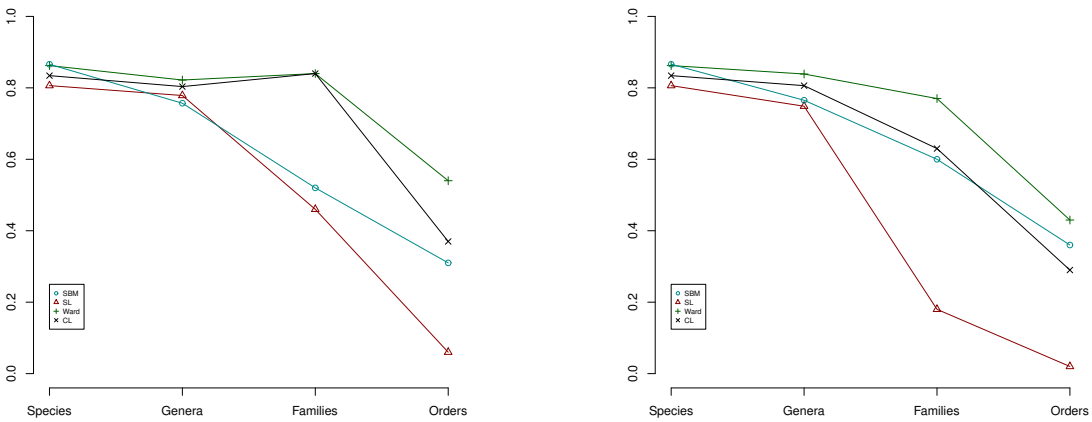


Figure 4: Adequacy between botanical and molecular-based classifications along a gradient of taxonomic levels, x axis: taxonomic levels y axis: Normalised Mutual Information (NMI) between molecular-based classification and botanical classification. One curve corresponds to one molecular-based classification. NMI are computed for classifications obtained of the samples (one per taxonomic level) presented in table 1. Left panel: Smith-Waterman dissimilarities. Right panel: kmer-based distances.

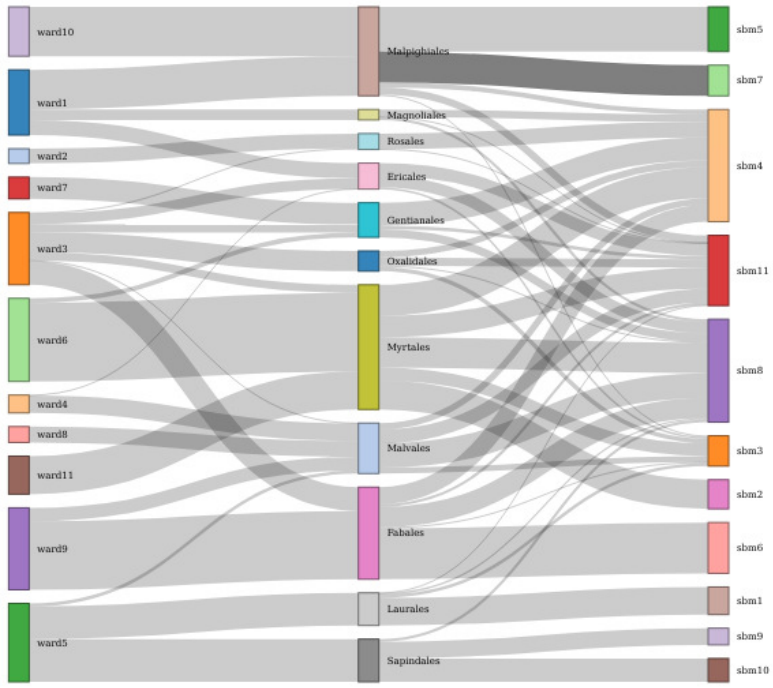


Figure 5: Sankey plot of correspondences between AHC with Ward (left column), botanical (central column) and SBM classification (right column) at the order level. The width of a flow between two classes is proportional to the number of sequences belonging to the two classes.

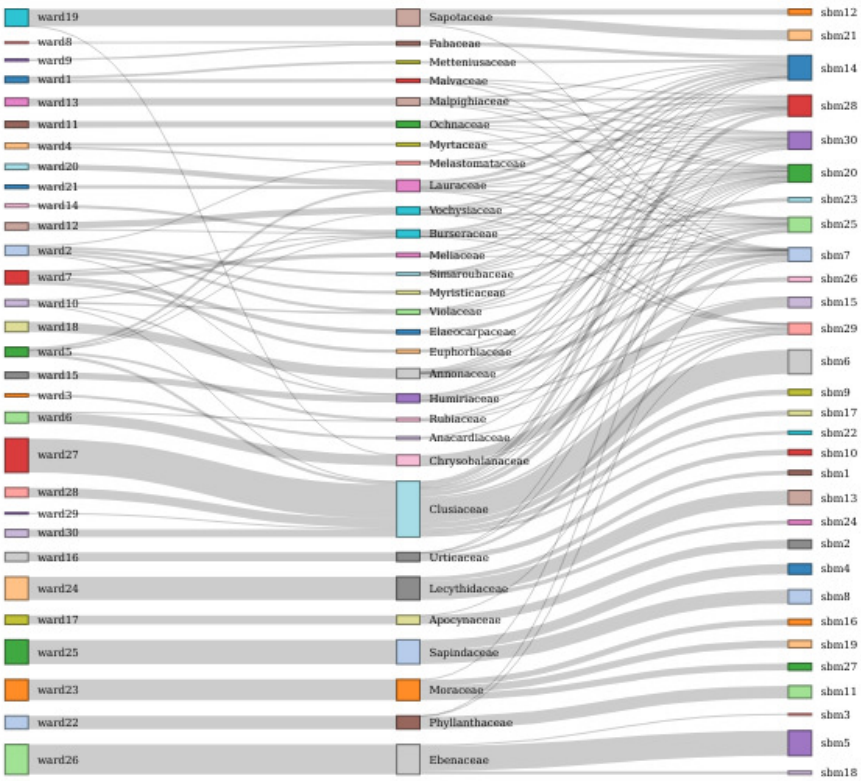


Figure 6: Sankey plot of correspondences between AHC with Ward (left column), botanical (central column) and SBM classification (right column) at the family level. The width of a flow between two classes is proportional to the number of sequences belonging to the two classes.

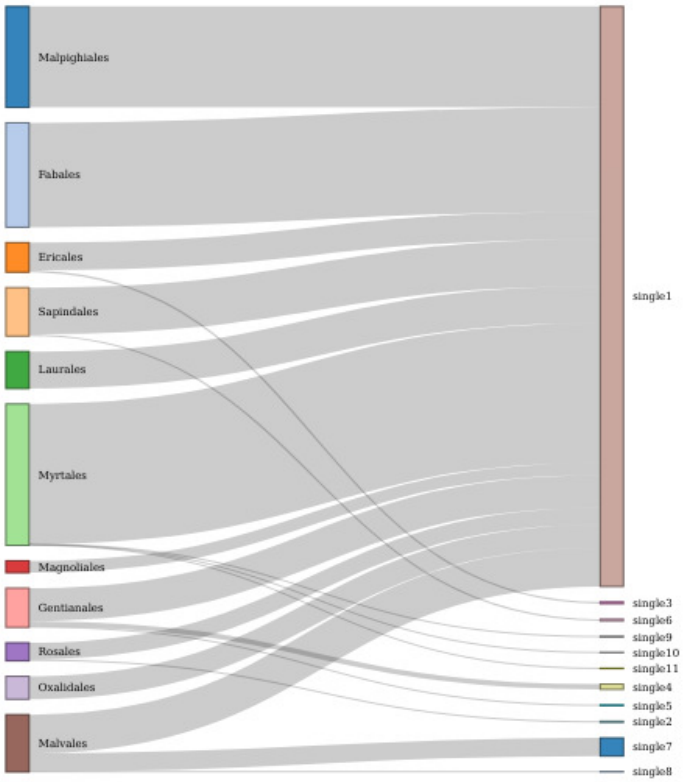


Figure 7: Sankey plot of correspondences between botanical classification (left column) and AHC with Single Linkage (right column), at the order level. The width of a flow between two classes is proportional to the number of sequences belonging to the two classes.

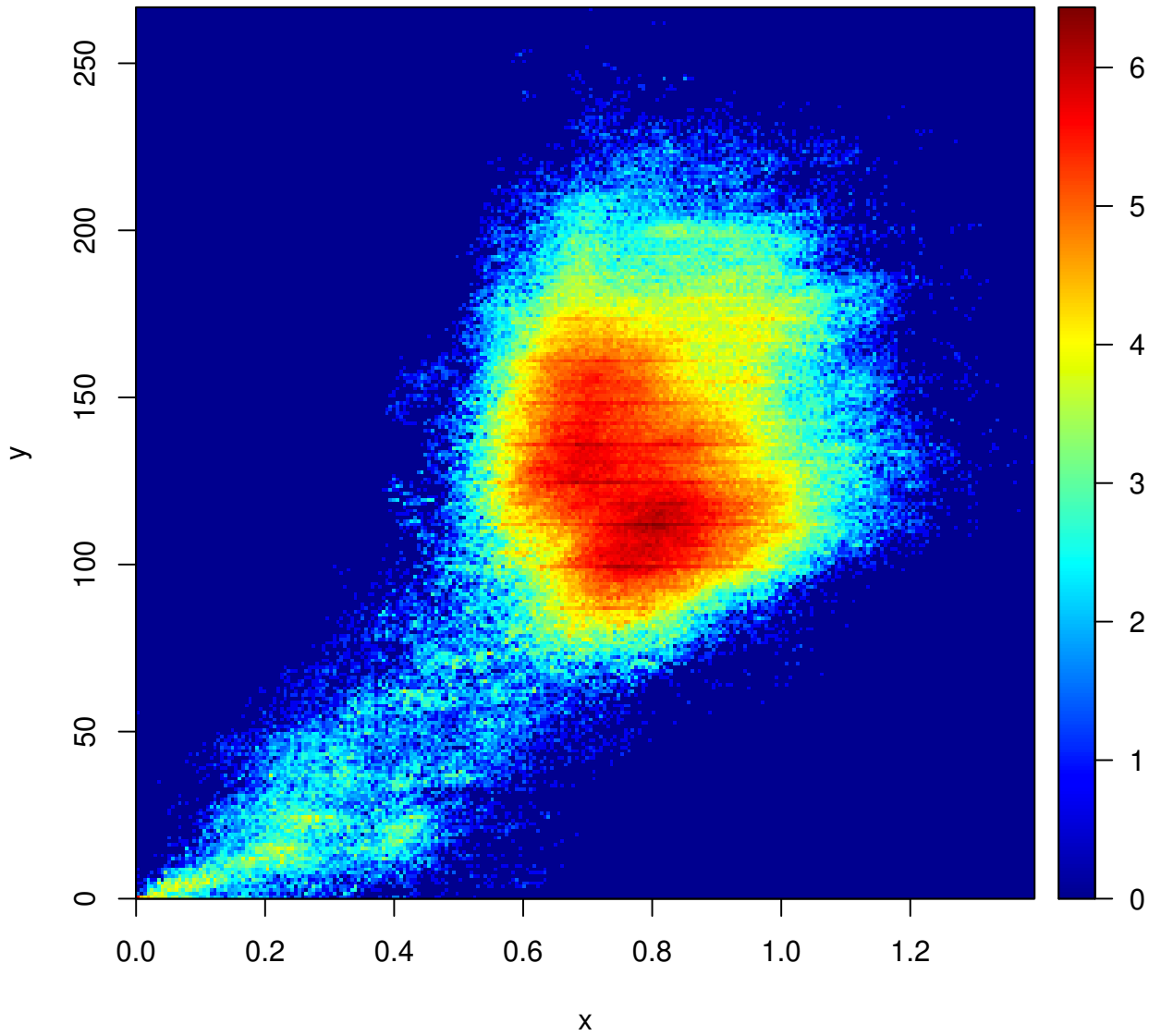


Figure 8: Comparison between Smith-Waterman and kmer-based dissimilarities (length $k = 4$). Density heatmap with logarithmic scale. x axis: kmer-based distance; y axis: Smith-Waterman dissimilarity. The color at a given pixel represents the logarithm of the number of pairs of sequences.

Bibliographie

[46108] Ieee standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, 2008.

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215 :403–410, 1990.

[And01] Christophe Andrieu. An Introduction to MCMC for Machine Learning. page 39, 2001.

[APF22] Mohamed Anwar Abouabdallah, Nathalie Peyrard, and Alain Franc. Does clustering of DNA barcodes agree with botanical classification directly at high taxonomic levels ? Trees in French Guiana as a case study. *Molecular Ecology Resources*, 22(5) :1746–1761, July 2022.

[AS66] S. M. Ali and SAMUEL D. Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the royal statistical society series b-methodological*, 28 :131–142, 1966.

[Bay13] Burak Bayramli. Svd factorization for tall-and-fat matrices on map/reduce architectures. *arXiv preprint arXiv :1310.4664*, 2013.

[BDLBH15] Pierre Barbillon, Sophie Donnet, Emmanuel Lazega, and Avner Bar-Hen. Stochastic block models for multiplex networks : an application to networks of researchers, 2015.

[BE99] Stephen P Borgatti and Martin G Everett. Models of corerperiphery structures. 1999.

[Bis06] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, 2006.

[BLW86] Norman Biggs, Lloyd, and Robin J. Wilson. *Graph theory, 1736-1936*. Clarendon Press, February 1986.

[BLZ16] Charles Bouveyron, Pierre Latouche, and Rawya Zreik. The stochastic topic block model for the clustering of vertices in networks with textual edges. *Statistics and Computing*, 2016.

[BMD20] H. R. Bhapkar, Parikshit N. Mahalle, and Prashant S. Dhotre. *Virus Graph and COVID-19 Pandemic : A Graph Theory Approach*, pages 15–34. Springer International Publishing, Cham, 2020.

[BRU67] STEPHEN G. BRUSH. History of the lenz-ising model. *Rev. Mod. Phys.*, 39 :883–893, Oct 1967.

[Cic14] Andrzej Cichocki. Tensor networks for big data analytics and large-scale optimization problems, 2014.

[CLBD22] Saint-Clair Chabert-Liddell, Pierre M Barbillon, and Sophie Donnet. Impact of the mesoscale structure of a bipartite ecological interaction network on its robustness through a probabilistic modeling. *Environmetrics*, 33(2), March 2022.

[CMS+19a] H. Caron, J.-F. Molino, D. Sabatier, P. Léger, P. Chaumeil, C. Scotti-Saintagne, J.-M. Frigério, I. Scotti, A. Franc, and Rémy J. Petit. Chloroplast DNA variation in a hyperdiverse tropical tree community. *Ecology and Evolution*, 9(8) :4897–4905, 2019.

[CMS+19b] Henri Caron, Jean-François Molino, Daniel Sabatier, Patrick Léger, Philippe Chaumeil, Caroline Scotti-Saintagne, Jean-Marc Frigério, Ivan Scotti, Alain Franc, and Rémy J. Petit. Chloroplast DNA variation in a hyperdiverse tropical tree community. *Ecology and Evolution*, 9(8) :4897–4905, March 2019.

[DBB20] Hussam Al Daas, Grey Ballard, and Peter Benner. Parallel algorithms for tensor train arithmetic. Technical Report 2011.06532, arXiv, 2020.

[DBC+21] Hussam Al Daas, Grey Ballard, Paul Cazeaux, Eric Hallman, Agnieszka Miedlar, Mirjeta Pasha, Tim W. Reid, and Arvind K. Saibaba. Randomized algorithms for rounding in the tensor-train format, 2021.

[DBNW20] Gaspard Ducamp, Philippe Bonnard, Anthony Nouy, and Pierre-Henri Wuillemin. An efficient low-rank tensors representation for algorithms in complex probabilistic graphical models. In Manfred Jaeger and Thomas Dyhre Nielsen, editors, *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 173–184. PMLR, 23–25 Sep 2020.

[DH19] Amelia Drew and Alexander Heinecke. Training neural machine translation (nmt) models using tensor train decomposition on tensorflow (t3f), 2019.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1) :1–38, 1977.

[DPR08] J.-J. Daudin, F. Picard, and S. Robin. A mixture model for random graphs. *Statistics and Computing*, 18(2) :173–183, June 2008.

[ER59] P. Erdös and A. Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6 :290, 1959.

[EY36] G. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychutnetiku*, (1) :211–218, 1936.

[FB19] Thorben Funke and Till Becker. Stochastic block models : A comparison of variants and inference methods. *PLOS ONE*, 14(4) :e0215296, April 2019.

[FCS+21] J.-M. Frigerio, H. Caron, D. Sabatier, J.-F. Molino, and A. Franc. Guiana Trees. 2021. Portail Data INRAE, V1, DOI : 10.15454/XSJ079.

[FH82] Ove Frank and Frank Harary. Cluster Inference by Using Transitivity Indices in Empirical Graphs. *Journal of the American Statistical Association*, 77(380) :835–840, 1982. Publisher : [American Statistical Association, Taylor & Francis, Ltd.].

[FHP91] Peter Fletcher, Hughes Hoyle, and C. Wayne Patty. *Foundations of Discrete Mathematics (International student ed.).* 1991.

[Fra22a] Alain Franc. Tensor ranks for the pedestrian for dimension reduction and disentangling interactions, 2022.

[Fra22b] Alain Franc. Tensor Ranks for the Pedestrian for Dimension Reduction and Disentangling Interactions, January 2022. arXiv :2201.07473 [cs, math, stat].

[FYZS18] Joshua Faskowitz, Xiaoran Yan, Xi-Nian Zuo, and Olaf Sporns. Weighted Stochastic Block Models of the Human Connectome across the Life Span. *Scientific Reports*, 8(1) :12997, December 2018.

[Gen98] James E. Gentle. Numerical linear algebra for applications in statistics. 1998.

[GVL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, USA, 1996.

[Hac19] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*, volume 56 of *Springer Series in Computational Mathematics*. Springer, second edition, 2019.

[HJ12] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge, second edition, 2012.

[HLL83] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels : First steps. *Social Networks*, 5(2) :109–137, 1983.

[Hub01] S.P. Hubbell. *The Unified Neutral Theory of Biodiversity and Biogeography*. Princeton University Press., 2001.

[JBO02] Pedro Jordano, Jordi Bascompte, and Jens M. Olesen. Invariant properties in coevolutionary networks of plant-animal interactions. *Ecology Letters*, 6 :69–81, 2002.

[JZS09] Qixia Jiang, Yan Zhang, and Maosong Sun. Community detection on weighted networks : A variational bayesian method. In Zhi-Hua Zhou and Takashi Washio, editors, *Advances in Machine Learning*, pages 176–190, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[KB09] T. G. Kolda and B. W. Bader. Tensor decomposition and applications. *SIAM Review*, 51(3) :455–500, 2009.

[KBCG12] Christine Keribin, Vincent Brault, Gilles Celeux, and Gérard Govaert. Model selection for the binary latent block model. In *20th International Conference on Computational Statistics (COMPSTAT 2012)*, pages 379–390, Limassol, Cyprus, August 2012.

[KBKK] Patrick Kalmbach, Andreas Blenk, Markus Kluegel, and Wolfgang Kellerer. Generating Synthetic Internet- and IP-Topologies using the Stochastic-Block-Model. page 6.

[KN11] Brian Karrer and M. E. J. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1) :016107, January 2011.

[LBA11] Pierre Latouche, Etienne Birmelé, and Christophe Ambroise. Overlapping stochastic block models with application to the french political blogosphere. *The Annals of Applied Statistics*, 5(1) :309–336, 2011.

[LBA12] Pierre Latouche, Etienne E. Birmelé, and Christophe Ambroise. Variational Bayesian Inference and Complexity Control for Stochastic Block Models. *Statistical Modelling*, 12(1) :93–115, 2012.

[LC15] Namgil Lee and Andrzej Cichocki. Estimating a few extreme singular values and vectors for large-scale matrices in tensor train format. *SIAM Journal on Matrix Analysis and Applications*, 36(3) :994–1014, jan 2015.

[LC18] Namgil Lee and Andrzej Cichocki. Fundamental tensor operations for large-scale data analysis using tensor network formats. *Multidimensional Syst. Signal Process.*, 29(3) :921–960, jul 2018.

[Leg16] Jean-Benoist Leger. Blockmodels : A r-package for estimating in latent block model and stochastic block model, with various probability functions, with or without covariates, 2016.

[Les06] Annick Lesne. Complex Networks : from Graph Theory to Biology. *Letters in Mathematical Physics*, 78(3) :235–262, December 2006.

[LK15] M. Leray and N. Knowlton. DNA barcoding and metabarcoding of standardized samples reveal patterns of marine benthic diversity. *PNAS*, 112(7) :2076–2081, 2015.

[LMZ+20] Chunxiao Liu, Zhendong Mao, Tianzhu Zhang, Hongtao Xie, Bin Wang, and Yongdong Zhang. Graph structured network for image-text matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[LR13] Pierre Latouche and Stéphane Robin. Estimation de la fonction graphon d'un W-graphe. Application au réseau de la blogosphere politique française. In *45èmes journées de Statistique*, Actes des 45èmes journées de Statistique, page JdS, Toulouse, France, 2013.

[LW19] C. Lee and D.J. Wilkinson. A review of stochastic block models and extensions for graph clustering. *Applied Network Science*, 4(122), 2019.

[MB09] D. König Michael and Stefano Battiston. *From Graph Theory to Models of Economic Networks. A Tutorial*, pages 23–63. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[MIR60] L. MIRSKY. SYMMETRIC GAUGE FUNCTIONS AND UNITARILY INVARIANT NORMS. *The Quarterly Journal of Mathematics*, 11(1) :50–59, 01 1960.

[MRV10] Mahendra Mariadassou, Stéphane Robin, and Corinne Vacher. Uncovering latent structure in valued graphs : A variational approach. *The Annals of Applied Statistics*, 4(2) :715 – 742, 2010.

[Mur83] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4) :354–359, 1983.

[Mü13] D. Müllner. fastcluster : Fast Hierarchical, Agglomerative Clustering Routines for R and Python. *Journal of Statistical Software*, 53(9) :1–18, 2013.

[NIK+20] Alexander Novikov, Pavel Izmailov, Valentin Khrulkov, Michael Figurnov, and Ivan Oseledets. Tensor train decomposition on tensorflow (t3f). *Journal of Machine Learning Research*, 21(30) :1–7, 2020.

[Nis09] Martin Niss. History of the lenz-ising model 1950–1965 : from irrelevance to relevance. *Archive for History of Exact Sciences*, 63(3) :243–287, 2009.

[NLTK19] Zhenguo Nie, Roby Lynn, Tommy Tucker, and Thomas Kurfess. Voxel-based analysis and modeling of mrr computational accuracy in milling process. *CIRP Journal of Manufacturing Science and Technology*, 27 :78–92, 2019.

[NPOV15a] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing Neural Networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[NPOV15b] Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry Vetrov. Tensorizing neural networks, 2015.

[NROV14] Alexander Novikov, Anton Rodomanov, Anton Osokin, and Dmitry Vetrov. Putting MRFs on a Tensor Train. *Proceedings of The 31st International Conference on Machine Learning*, 2014.

[NS01a] Krzysztof Nowicki and Tom A. B. Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96 :1077 – 1087, 2001.

[NS01b] Krzysztof Nowicki and Tom A. B Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455) :1077–1087, 2001.

[OD12] I. V. Oseledets and S. V. Dolgov. Solution of linear systems and matrix inversion in the tt-format. *SIAM Journal on Scientific Computing*, 34(5) :A2718–A2739, 2012.

[Orú14] Román Orús. A practical introduction to tensor networks : Matrix product states and projected entangled pair states. *Annals of Physics*, 349 :117–158, oct 2014.

[Orú19] Román Orús. Tensor networks for complex quantum systems. *Nature Reviews Physics*, 1(9) :538–550, aug 2019.

[Ose09] I. V. Oseledets. A new tensor decomposition. *Dokl. Math.*, 80(1) :495–496, 2009.

[Ose10] I. V. Oseledets. Approximation of $2^d \times 2^d$ matrices using tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4) :2130–2145, 2010.

[Ose11] I V Oseledets. TENSOR-TRAIN DECOMPOSITION. page 24, 2011.

[OTZ11] Ivan Oseledets, Eugene Tyrtyshnikov, and Nickolai Zamarashkin. Tensor-Train Ranks for Matrices and Their Inverses. page 10, 2011.

[PLP09] D. Pfitzner, R. Leibbrandt, and D. Powers. Characterization and evaluation of similarity measures for pairs of clusterings. *Knowledge and Information Systems*, 19(361), 2009.

[RC04] C.P. Robert and G. Casella. *Monte Carlo statistical methods (second edition)*. Springer Verlag, 2004.

[Sni97] Snijders, Tom A. B. and Nowicki, Krzysztof. Estimation and Prediction for Stochastic Blockmodels for Graphs with Latent Block Structure. 14 :75–100, 1997.

[SST03] Eric M. Schwarz, Martin Schmookler, and Son Dao Trong. Hardware implementations of denormalized numbers. In *Proceedings of the 16th IEEE Symposium on Computer Arithmetic (ARITH-16'03)*, ARITH '03, page 70, USA, 2003. IEEE Computer Society.

[Ste93] G. W. Stewart. On the early history of Singular Value Decomposition. *SIAM review*, 35(4) :551–566, 1993.

[Str19] G. Strang. *Linear Algebra and Learning from Data*. Wellesley Cambridge Press, 2019.

[SW81] P. D. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147 :195–197, 1981.

[Syl] James Sylvester. Chemistry and algebra. *Nature*, 17 :284–284.

[TBC17] Timothée Tabouy, Pierre Barbillon, and Julien Chiquet. Variational Inference for Stochastic Block Models from Sampled Data. working paper or preprint, September 2017.

[TL10] Lei Tang and Huan Liu. *Graph Mining Applications to Social Network Analysis*, pages 487–513. Springer US, Boston, MA, 2010.

[Wis69] David Wishart. 256. note : An algorithm for hierarchical classifications. *Biometrics*, 25(1) :165–170, 1969.

[WRR03] Michael E. Wall, Andreas Rechtsteiner, and Luis M. Rocha. *Singular Value Decomposition and Principal Component Analysis*, pages 91–109. Springer US, Boston, MA, 2003.

[WW87] Yuchung J. Wang and George Y. Wong. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82(397) :8–19, March 1987.

[YFW03] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millennium*, chapter 8, pages 239–236. Morgan Kaufmann Publishers, January 2003.

[ZYO+15] Zheng Zhang, Xiu Yang, Ivan V. Oseledets, George E. Karniadakis, and Luca Daniel. Enabling high-dimensional hierarchical uncertainty quantification by anova and tensor-train decomposition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(1) :63–76, 2015.