UNIVERSITÉ DU QUÉBEC À MONTRÉAL

D-DGC : UNE APPROCHE DYNAMIQUE POUR LE CLUSTERING PROFOND DES GRAPHES ATTRIBUÉS

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR
MOHAMED FAWZI TOUATI

AOÛT 2022

REMERCIEMENTS

Je tiens à exprimer toute ma reconnaissance à mon directeur de recherche, Professeur Mohamed BOUGUESSA. Je le remercie de m'avoir encadré, orienté, aidé et conseillé, et pour la liberté de travail qu'il m'a laissée tout au long de ce programme. Je le remercie aussi pour sa disponibilité, sa patience et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion.

Je remercie également la faculté des sciences de l'UQAM pour l'octroi de la bourse d'exemption des frais de scolarité majorés que j'ai reçue durant ma maîtrise. Je remercie aussi, l'organisme Mitacs de m'avoir offert l'opportunité d'un stage de recherche au sein de l'entreprise Ciena. J'adresse également ma gratitude à la Fondation de l'UQAM, pour m'avoir choisi parmi les récipiendaires d'une bourse de soutien à la réussite de la Maîtrise en informatique.

J'adresse mes sincères remerciements à tous les professeurs de l'UQAM, intervenants, et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques, ont guidé mes réflexions et ont accepté de me rencontrer, et répondre à mes questions durant mon cursus et mes recherches, et qui doivent voir dans ce travail la fierté d'un savoir bien acquis. J'exprime, enfin, mes plus vifs remerciements à mes collègues au sein du laboratoire LATECE, en particulier, Etienne, Fares et Nairouz.

Je remercie mes très chers parents, qui ont toujours été là pour moi. Ma mère, qui a œuvré pour ma réussite, de par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie : reçois à travers ce travail, aussi modeste soit-il, l'expression de mes sentiments

et de mon éternelle gratitude. Mon père, qui peut être fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Merci pour les valeurs nobles, l'éducation et le soutien permanent venu de toi.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

DÉDICACE

Je dédie ce travail à :

Mes parents. Aucun hommage ne pourrait être à la hauteur de l'amour dont ils ne cessent de me combler.

Toute ma famille, tous mes amis, et à tous ceux qui ont contribué de près ou de loin pour que ce travail soit possible.

Je vous dis tous, merci.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

RÉSUMÉ

L'analyse des graphes attribués est un domaine de l'exploration de données qui a attiré l'attention des chercheurs ces dernières années. En plus de la représentation classique des graphes où deux nœuds sont reliés par un lien, deux nœuds dans un graphe attribué possèdent un ensemble de caractéristiques déterminantes. De ce fait, une des problématiques fondamentales étudiées dans ce domaine est le clustering des graphes attribués, où chaque nœud du graphe est affecté à un cluster, selon des critères d'appartenance bien définis.

Malgré que le clustering des graphes suscite l'intérêt dans le contexte non attribué, les graphes attribués donnent un formalisme intéressant à explorer dans la recherche, du fait de l'incorporation des attributs des nœuds et les caractéristiques de ces derniers ; ce qui donne davantage d'information à explorer pour la tâche de clustering. L'évolution des techniques de clustering de son côté a permis de découvrir des approches dites profondes basées essentiellement sur les autoencodeurs de graphes. Le clustering profond utilise une fonction de coût, celle-ci doit être minimisée pour optimiser la reconstruction faite par les autoencodeurs, et le clustering, selon la méthode de combinaison choisie. Bien qu'il existe un bon nombre de travaux abordant cette problématique, certains aspects demeurent peu ou pas abordés dans la littérature. Notamment : (1) l'aspect statique de la fonction de coût des modèles de clustering des graphes attribués, et (2) la sensibilité des modèles profonds au caractère arbitraire des descripteurs qui sont générés par les modèles d'apprentissage profond.

Afin de pallier les limites des approches existantes, nous présentons dans le cadre de ce mémoire une nouvelle approche de clustering profond des graphes attribués. En exploitant le principe de l'apprentissage dynamique, l'approche proposée combine la fonction de coût d'un modèle basé sur les autoencodeurs de graphes à un paramétrage dynamique permettant une meilleure séparation des clusters grâce à un critère de confiance. L'efficacité de l'approche proposée est comparée à d'autres méthodes récentes sur différents ensembles de données. Les résultats obtenus démontrent l'efficacité de notre approche à diminuer l'impact du caractère arbitraire des descripteurs dans le cadre du clustering profond des graphes attribués.

Mots clés : Graphes attribués, Clustering profond, Apprentissage dynamique.

CHAPITRE I

INTRODUCTION

1.1 Mise en contexte

Les graphes sont des structures de données très utilisées, et un langage universel pour décrire des systèmes du monde réel. Formellement, un graphe $\mathcal{G} = (V, E)$ (décrit dans la figure 1.1) est une structure formée de deux ensembles de données. Le premier ensemble (V) constitue les entités (nœuds) du graphe, et le deuxième ensemble (E) constitue les interactions (liens) entre les entités du graphe. On note un lien allant du nœud $u \in V$ au nœud $v \in V$ par $(u, v) \in E$.

Figure 1.1: Visualisation la structure d'un graphe.

Ainsi, les graphes offrent une base mathématique sur laquelle les chercheurs s'appuient pour analyser, comprendre, et apprendre des systèmes complexes du monde réel. Pour cela, beaucoup de données structurées sous forme de graphes ont été mises à disposition des chercheurs (Song *et al.*, 2005). Ceci est dû à la puissance du formalisme des graphes qui permet de se focaliser sur les relations entre les entités (plutôt que les propriétés des entités uniquement).

Ce paradigme existe dans différents domaines, à titre d'exemple, un réseau routier peut être représenté sous forme de graphe (El Mahrsi et Rossi, 2012), dans ce cas, il est possible d'utiliser un ensemble de nœuds pour décrire les villes d'un réseau, et les liens pour désigner les liaisons routières entre ces villes. Dans un autre contexte, pour refléter un réseau social sous forme de graphe (Singh *et al.*, 2016), il est possible d'utiliser un ensemble de nœuds pour illustrer les individus d'un réseau, et les liens pour indiquer la relation amicale entre l'ensemble d'individus. Dans le domaine biologique (Gao *et al.*, 2018), l'ensemble de nœuds correspond aux molécules (exemple : ADN ou ARN), et l'ensemble des liens représente la relation entre eux. Nous pourrions aussi utiliser les graphes pour produire des bases de données d'appareils connectés au Web (Yao *et al.*, 2013), où les nœuds constituent des objets, et les liens évoquent la relation entre eux. Ainsi, le défi de l'utilisation des graphes consiste à exploiter et modéliser les données réelles.

Au-delà de la définition pure de ce qu'est un graphe, on retrouve de nombreux types de graphes tels que les graphes hétérogènes (Pan *et al.*, 2015), les graphes multiplex (Heaney, 2014) et les graphes attribués (Combe *et al.*, 2015). Le présent se focalise principalement sur les graphes attribués.

En effet, dans de nombreux cas, on retrouve des informations sur les attributs, ou les caractéristiques associées à un graphe (exemple : des informations sur le profil associé à un utilisateur dans un réseau social). C'est ce qui caractérise les graphes attribués. Le plus souvent, il s'agit d'attributs au niveau du nœud, ils sont représentés à l'aide d'une valeur réelle dans la matrice d'adjacence $\boldsymbol{A} = \{a_{ij}\} \in \mathbb{R}^{n \times n}$ de $\mathcal{G}$, avec $a_{i,j} = 1$ si $(v_i, v_j) \in \boldsymbol{E}$, sinon $a_{i,j} = 0$, où l'on suppose que l'ordre des nœuds est similaire dans la matrice de caractéristiques des attributs. La figure ci-dessous présente un exemple de la structure d'un graphe attribué, contenant un ensemble de nœuds. Chaque nœud dans le graphe possède un ensemble de caractéristiques.

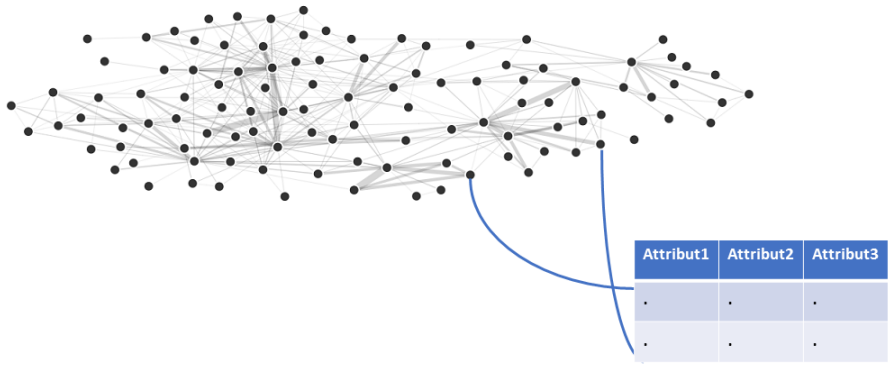


Figure 1.2: Image reflétant la structure d'un graphe attribué.

Ainsi, notre intérêt d'étudier les graphes attribués réside dans l'incorporation des données des nœuds et leurs caractéristiques. En effet, nous allons voir dans ce travail comment les attributs peuvent être modélisés, puis utilisés avec des informations structurelles dans l'analyse des graphes.

Une des tâches fondamentales de l'analyse des graphes est le clustering, celui-ci permet de partitionner les nœuds du graphe en sous-graphes disjoints. En pratique, le clustering de graphes joue un rôle déterminant dans plusieurs applications d'extraction de graphes telle que la détection de communauté (Fortunato, 2010). Si on prend l'exemple d'un réseau de chercheurs, et nous faisons un graphe de collaboration qui relie deux chercheurs s'ils ont co-rédigé un article ensemble. Dans ce cas, si on examine ce graphe de manière triviale, on aura une analyse qui ne distinguera pas les groupes de chercheurs. Donc, il est plus probable que le graphe se divise en différents groupes de nœuds, regroupés par domaine de recherche, institution, ou d'autres facteurs démographiques. C'est l'intuition générale de la tâche de clustering des graphes, dont le défi est de déduire des structures communautaires latentes étant donné uniquement le graphe d'entrée $\mathcal{G} = (V, E)$.

De nombreuses applications réelles du clustering comprennent la découverte de modules fonctionnels dans les réseaux d'interaction génétique (Talwar *et al.*, 2014), et la découverte de groupes d'utilisateurs frauduleux dans les réseaux de transactions financières (Palshikar et Apte, 2008).

Le défi du clustering de graphes est de savoir comment définir des fonctionnalités utiles qui prennent en compte la structure relationnelle au sein de chaque nœud. En effet, les relations par paires de nœuds sont plus exploitables pour la tâche de clustering, ceci est dû aux fonctionnalités des nœuds, et des arêtes qui capturent la relation sémantique entre les nœuds. Alors que pour les représentations des attributs des nœuds, une structure multiple doit être apprise pour capturer les similarités de haut niveau. Dans plusieurs scénarios, la matrice d'adjacence du graphe et les attributs des nœuds contiennent des informations complémentaires. Dans ce qui suit, nous présentons une visualisation d'un graphe attribué avec les clusters de nœuds colorés.

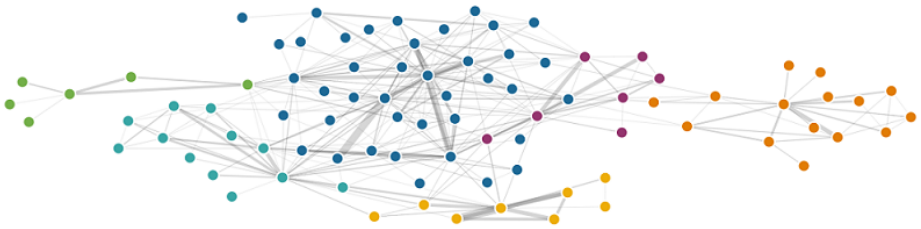


Figure 1.3: Visualisation des clusters dans un graphe.

L'autre volet d'intérêt de notre recherche se base sur les nouvelles techniques d'apprentissage automatique. En effet, l'apprentissage automatique est par nature une discipline axée sur les problèmes. Nous cherchons à construire des modèles, qui peuvent apprendre à résoudre des tâches particulières (exemple : le traitement de textes, ou reconnaissance des formes) à partir des données fournies en entrée. Les modèles d'apprentissage peuvent être, en général, de type supervisé ou non supervisé.

L'apprentissage automatique sur les graphes n'est pas si différent que dans les autres domaines, on retrouve les catégories habituelles d'apprentissage supervisé et non supervisé. Les problèmes supervisés sont populaires avec les données de graphes telles que la classification des nœuds (Tang *et al.*, 2016), mais les problèmes non supervisés suscitent eux aussi un intérêt majeur tels que le clustering des nœuds ou des graphes (Firat *et al.*, 2007).

D'autre part, le progrès connu par l'apprentissage profond dans les domaines de la classification de larges données, dans les disciplines telles que la vision par ordi-

nateur (Brunetti *et al.*, 2018), et le traitement automatique de la langue naturelle (Young *et al.*, 2018) ont motivé son développement dans le domaine non supervisé. Pour généraliser cette réalisation au contexte non supervisé, le clustering profond (Guo *et al.*, 2017; Caron *et al.*, 2018) est apparu comme une technique permettant d'effectuer un apprentissage conjoint des fonctionnalités, et un clustering intégré à l'aide de réseaux de neurones profonds. Ce dernier a montré un grand potentiel pour résoudre des problèmes de clustering complexes (Xie *et al.*, 2016).

La plupart des méthodes de clustering profond combinent la pseudosupervision et l'autosupervision pour surmonter l'absence des signaux de supervision. D'une part, la pseudosupervision encourage l'apprentissage de représentations orientées clustering. Elle consiste à « guider » un modèle d'apprentissage avec des pseudoétiquettes, qui sont généralement extraites en appliquant une méthode de clustering standard sur les représentations latentes (*embeddings*). En raison de sa nature non supervisée, certaines des pseudoétiquettes obtenues ne correspondent pas aux vrais clusters. Par conséquent, la pseudosupervision peut détériorer les performances du clustering.

D'autre part, l'autosupervision encourage l'apprentissage des descripteurs non orientés au clustering. Plus particulièrement, l'autosupervision consiste à entraîner un réseau de neurones pour résoudre une tâche prétexte [1], celle-ci exige une compréhension sémantique de haut niveau des données d'entrée. En pratique, l'autosupervision joue deux rôles importants dans le contexte du clustering profond. Tout d'abord, elle est utilisée pour préentraîner le modèle, afin d'éviter l'extraction

1. Une tâche prétexte est une étape de préentraînement d'un modèle d'apprentissage profond. Celle-ci est effectuée avant l'étape d'apprentissage finale du modèle, dans le but d'extraire des représentations latentes utiles, remplaçant ainsi le manque de données étiquetées en entrée. On peut citer l'exemple de la colorisation des images, où le modèle prédit la version couleur d'une image en fonction de sa version en niveaux de gris, et extrait les représentations latentes pertinentes pour la détection des formes.

de pseudoétiquettes basées sur des descripteurs dans l'espace des représentations. Ensuite, elle est utilisée pendant le processus du clustering, pour atténuer l'impact négatif de l'application de pseudoétiquettes erronées.

La conception architecturale des méthodes de clustering profond existantes dépend de la technique d'autoencodeur (Baldi, 2011) choisie. Deux principaux modes autosupervisés ont été largement utilisés dans la littérature sur le clustering profond. Le premier consiste à reconstruire les nœuds du graphe en entrée après les avoir projetés dans un espace latent de faible dimension. Le second assure la cohérence entre les assignations des points de données transformés, et les clusters identifiés des points de données réels. Cependant, apprendre des identifications cohérentes entre les données d'entrée A et les données transformées $f(A)$ nécessite des connaissances préalables en ce qui concerne les transformations de données possibles f, qui peuvent ne pas être disponibles pour certains ensembles de données.

Dans le cas d'un clustering profond sur les graphes attribués, les techniques de transformations de données ne sont pas claires, car elles devraient être appliquées aux entités de nœud de manière à obtenir des identifications cohérentes. Cela explique pourquoi l'autoencodeur est la conception architecturale la plus utilisée pour le clustering profond des graphes attribués.

Bien que les méthodes traditionnelles de clustering de graphes aient eu un impact significatif, en combinant des fonctionnalités de contenu avec des informations structurées en graphes, les techniques récentes utilisent une stratégie d'incorporation de nœuds prometteuse, appelée réseaux de neurones convolutifs de graphes (GCN) (Kipf et Welling, 2017). Les GCNs bénéficient du pouvoir de représentation de l'apprentissage profond, c'est-à-dire que les caractéristiques de chaque couche sont obtenues en multipliant la sortie de la couche précédente par une matrice de

poids, puis filtrées en utilisant une variante efficace de convolutions de graphe. L'intuition de l'opération convolutive de graphe est d'exploiter la structure du graphe en lissant les caractéristiques de contenu de chaque nœud sur son voisinage. Motivés par les GCNs (Kipf et Welling, 2017), l'autoencodeur de graphes (GAE), et l'autoencodeur variationnel de graphes (VGAE) (Kipf et Welling, 2016) ont montré des réalisations notables dans plusieurs applications de clustering de graphes attribués. En plus des approches basées sur GAE, qui optimisent les fonctions de coût statiques pendant tout le processus de clustering, la méthode que nous proposons dans ce mémoire exploite le cadre de l'apprentissage dynamique.

L'apprentissage dynamique (Mrabah *et al.*, 2020) est un paradigme non supervisé, inspiré des systèmes dynamiques naturels. Il suggère de remédier à l'absence de signaux de supervision en convertissant progressivement une fonction de coût [2] autocontrôlée à usage général en une fonction pseudosupervisée spécifique à une tâche précise. Les réseaux de neurones dynamiques (Han *et al.*, 2021), par opposition aux réseaux de neurones statiques, peuvent adapter leurs structures, fonction de coût, ou paramètres à l'entrée ou lors de l'inférence, et donc profiter de propriétés favorables qui sont absentes dans les modèles statiques. En général, le calcul dynamique dans le contexte de l'apprentissage profond présente les avantages suivants :

1. **Efficacité :** il permet d'améliorer les architectures d'apprentissage profond statiques pour donner de meilleurs résultats.
2. **Pouvoir de représentation :** il a un espace de paramètres considérablement agrandi et amélioré, en raison de la dépendance des données aux architectures, ou aux paramètres du réseau.

2. Dans ce mémoire les termes « fonction de coût », « fonction objectif », et « fonction de perte » vont être utilisés d'une façon interchangeable afin de désigner la fonction à minimiser ou à maximiser pour déterminer la solution optimale à un problème de clustering.

3. **Adaptabilité :** il est capable d'atteindre un compromis souhaité entre précisions et efficacité pour gérer les coûts de calculs.
4. **Compatibilité :** il est compatible avec la majorité des architectures d'apprentissage profond.
5. **Généralité :** il peut être appliqué de manière transparente à un large éventail d'applications dans l'apprentissage profond.

1.2 Motivations

L'intérêt de ce travail porte sur le clustering profond des graphes attribués. Cette discipline connaît des avancées majeures grâce aux développements des GCN (Kipf et Welling, 2017) et des VGAE (Kipf et Welling, 2016). Certaines méthodes se sont inspirés des GCN, tel que le MGAE (Wang *et al.*, 2017a) qui utilise des représentations de nœuds avec un GCN à trois couches, puis applique un autoencodeur de débruitage (Chen *et al.*, 2012) pour reconstruire les caractéristiques des nœuds. Le clustering spectral (Ng *et al.*, 2002) est ensuite utilisé pour déterminer les clusters. On retrouve aussi les méthodes basées sur les GAE et VGAE, qui apprennent les représentations latentes (*embeddings*) des nœuds, puis ensuite, utilisent un modèle génératif GAN (Goodfellow *et al.*, 2020) pour renforcer une distribution a priori au niveau de l'espace latent (Pan *et al.*, 2018). Enfin, on retrouve le modèle DAEGC (Wang *et al.*, 2019), qui se base sur le mécanisme d'attention et VGAE, pour identifier des structures latentes permettant de dissocier les différents clusters.

En dépit de ces travaux, le clustering des graphes attribués continue de poser des défis aux méthodes existantes. En fait, les approches existantes citées précédemment souffrent des problèmes suivants :

1. L'aspect statique des fonctions de coût des modèles basés sur l'apprentissage profond ne permet pas d'explorer des améliorations potentielles ac-

quises dans les domaines connexes grâce à l'apprentissage dynamique.

2. Le caractère arbitraire des descripteurs générés dans l'espace de représentation. Ce problème se produit lorsqu'une quantité considérable de pseudoétiquettes générées ne correspond pas aux clusters réels. Une erreur cumulative infligée par la pseudosupervision peut rapidement dégrader les performances du modèle de clustering et ceci peine à être résolu.

1.3 Contributions

Afin de pallier les limitations constatées dans les modèles cités dans la section précédente, nous proposons dans le cadre de ce mémoire une nouvelle approche de clustering des graphes attribuées en utilisant l'apprentissage dynamique.

les principales contributions de ce travail sont :

1. Proposition d'une solution basée sur l'apprentissage profond permettant de trouver les caractéristiques déterminantes de chaque cluster dans le graphe attribué.
2. Proposition d'une solution permettant de diminuer l'impact du caractère arbitraire des descripteurs générés dans l'espace de représentation, durant le processus de clustering profond des graphes attribués.
3. Amélioration des performances du clustering profond des graphes attribués basés sur un autoencodeur de graphes. Les résultats obtenus montrent, toutefois, l'importance de réduire le caractère arbitraire des descripteurs générés dans l'espace de représentation, pour le clustering basé sur les autoencodeurs de graphes.

1.4 Structure et organisation du mémoire

Le chapitre 2 comportera la revue des travaux existants dans le clustering des graphes attribués, en pointant les méthodes traditionnelles et les méthodes modernes dites automatiques ou profondes. Pour chaque algorithme, une revue des motivations de recherche, des avantages, et des inconvénients sera explicitée. Ensuite, le chapitre 3 comportera la description de l'approche proposée, c'est-à-dire les deux phases de la solution D-DGC (Dynamic Deep Graph Clustering).

Afin de bien argumenter la solution proposée, le chapitre 4 comportera une étude détaillée de notre approche, et un protocole expérimental complet. Ce dernier sera dédié à l'évaluation de l'approche proposée sur des données de références, une comparaison avec les données des travaux récents et une évaluation complète des performances. En dernier, le chapitre 5 conclut le mémoire et exposera les perspectives du présent travail.

CHAPITRE II

REVUE DE LA LITTÉRATURE

Ce chapitre présente une revue de la littérature du clustering des graphes attribués. Comme il existe de nombreuses approches proposées, nous allons retenir celles ayant reçu le plus d'intérêt de la part de la communauté scientifique. À travers ces approches, nous décrirons une vue d'ensemble des techniques proposées selon leurs principes méthodologiques. Nous commencerons par présenter les méthodes de clustering de graphes attribués basées sur des métriques et des heuristiques bien définies. Ensuite, nous passerons en revue les méthodes basées sur des modèles d'apprentissage automatique et d'apprentissage profond. Celles-ci ont contribué au développement du clustering profond dans les graphes, notamment les techniques des réseaux de neurones convolutifs pour les graphes (GCN) et les autoencodeurs de graphes (GAE et VGAE).

Après cela, il sera question d'évoquer le principe de l'apprentissage profond dynamique, avec des exemples de modèles basés sur les architectures dynamiques, et le paramétrage dynamique. Enfin, nous évoquerons le travail existant, pour résoudre la problématique du caractère arbitraire des descripteurs dans le contexte du clustering profond. Dans ce qui suit, nous allons porter notre attention sur des travaux représentatifs dans chaque groupe d'algorithmes.

2.1 Approches traditionnelles du clustering des graphes attribués

On distingue trois approches différentes : (1) celles basées sur les algorithmes de marche aléatoire, (2) celles basées sur l'inférence statistique, et (3) celles basées sur les critères de dimensionnalité.

2.1.1 Méthodes basées sur la marche aléatoire

Une marche aléatoire sur un graphe éventuellement infini est un processus stochastique, où un marcheur se déplace de nœud en nœud, en choisissant un voisin cible au hasard à chaque étape (Noh et Rieger, 2004). Dans le contexte du clustering, les modèles de marche aléatoire sont utilisés pour estimer les distances entre les nœuds sur les graphes attribués. En fonction de cette distance, les approches de type k-moyennes (K-Means) attirent les nœuds autour des k-centroïdes prédéfinis afin de capturer les membres des clusters (Zhou *et al.*, 2009). Cette approche démontre que plus il y a de valeurs d'attributs partagées par deux nœuds, plus il y a de chemins via des nœuds d'attributs communs existants. De cette façon, les marches aléatoires peuvent être utilisées pour mesurer le nœud de proximité à travers les liens structurels (liens qui relient les nœuds d'un graphe) et les liens compositionnels (liens entre les vecteurs d'attributs associés aux nœuds).

Dans la méthode Connected K-Centers [3] (Ge *et al.*, 2008), la stratégie de marche aléatoire adoptée est une recherche simple en largeur (BFS : Breadth First Search), définie pour les graphes, où le vecteur d'attributs associé à un nœud est également utilisé pour déterminer le prochain nœud à visiter. La méthode citée utilise l'algorithme K-Means combiné aux marches aléatoires afin de calculer les distances

3. Dans ce mémoire, nous utilisons la nomination principale des modèles de clustering, et de clustering profond en anglais, telles qu'elles sont indiquées dans la littérature étudiée.

entre les nœuds ; en choisissant d'abord k nœuds aléatoires comme centres de cluster. Ensuite, tous les nœuds sont graduellement affectés à l'un des k clusters en traversant le graphe avec une recherche simple en largeur (BFS). Enfin, les centroïdes des clusters sont recalculés pour déterminer les nouveaux centres. Les deux dernières étapes sont répétées jusqu'à ce qu'il n'y ait plus de changement dans les centres des clusters.

2.1.2 Méthodes basées sur l'inférence statistique

Une inférence statistique (Tong, 2019) est une opération qui consiste à extraire les propriétés des ensembles de données, à partir d'un ensemble d'observations dans un modèle, puis à déduire des prédictions sur une population plus large. De la même façon, l'inférence statistique est utilisée dans le domaine du clustering des graphes attribués (Fortunato, 2010). Elle consiste à utiliser des modèles génératifs comme étape intermédiaire, afin de mélanger les vecteurs d'attributs associés aux nœuds, et la topologie du graphe dans un modèle unifié.

Une étude (Li *et al.*, 2008) propose une méthode de clustering, pour trouver des clusters dans un document regroupant un ensemble de textes à grande échelle. L'approche proposée exploite à la fois le contenu du document (les mots) et ses références ou citations. Elle utilise l'inférence statistique comme étape intermédiaire pour trouver des concepts latents et regrouper davantage de documents. Ainsi, l'objectif serait de rechercher les centres de chaque cluster, puis d'affecter graduellement chaque nœud (document texte) à son cluster (sujet ou thématique) correspondant. La détection des centres identifie les concepts latents des documents fréquemment co-référencés qui peuvent désigner des centres de clusters. La deuxième phase affecte les nœuds initiaux selon leur similarité thématique avec les centres de clusters. Enfin, la méthode Latent Dirichlet Allocation (LDA) (Blei

et al., 2003), utilisé fréquemment dans la fouille de textes, est appliqué pour le clustering des thématiques. LDA est un modèle génératif permettant à la fois de découvrir les mots qui appartiennent à un document, et les mots qui appartiennent à un sujet ou thématique. De ce fait, la probabilité d'appartenance de chaque mot de ces documents à une thématique est calculée. Enfin, l'étape suivante consiste à regrouper les documents restants aux clusters. LDA est également utilisé en tant qu'approche principale pour identifier les concepts latents (Liu *et al.*, 2009; Balasubramanyan et Cohen, 2011).

Le LDA Topic Link Model (Liu *et al.*, 2009) est un modèle qui prend en compte les thématiques, l'appartenance des auteurs, et l'existence de liens entre les paires de documents. Pour la tâche de clustering, l'approche citée précédemment offre un travail de recherche significatif, sur la façon dont la similitude du contenu et la similitude du cluster contribuent à la formation de liens. Celle-ci a été en mesure de révéler que l'adhésion des auteurs (inclus dans les documents textes) a un effet beaucoup plus fort sur la formation de liens entre les articles, aussi bien dans les domaines politiques, que dans les articles techniques. Ces travaux montrent également que la dimension thématique est plus importante que la similitude du cluster dans la citation d'articles. Dans le même domaine (Balasubramanyan et Cohen, 2011), le problème de la modélisation des liens en utilisant LDA a été également résolu.

Un autre travail (Xu *et al.*, 2012) propose un modèle de clustering qui se transforme en un problème d'inférence statistique. L'approche proposée définit un modèle bayésien génératif, qui produit un échantillon de toutes les combinaisons possibles d'un graphe. Ce graphe est défini par sa matrice d'adjacence $\mathbf{A}$, son vecteur d'attributs des nœuds $\mathbf{X}$, et un vecteur $\mathbf{Z}$ contenant l'assignation exacte de chaque nœud à un des k clusters. Ce modèle produit une probabilité conjointe $p(\mathbf{A}, \mathbf{X}, \mathbf{Z})$. L'idée est donc de trouver une partition $\mathbf{Z}^*$ tel que $\mathbf{Z}^* = \arg\max \mathbf{Z} p(\mathbf{Z} \mid \mathbf{A}, \mathbf{X})$.

Les techniques présentées dans cette section sont très intéressantes pour fusionner à la fois les attributs et la topologie dans le même modèle, mais malheureusement le processus d'optimisation pour estimer les paramètres de la vraisemblance est souvent coûteux. De plus, ils ne reposent sur la définition d'aucune distance, et le choix des distributions a priori dans les modèles statistiques nécessite une expertise non triviale.

2.1.3 Méthodes basées sur les critères de dimensionnalité

Certaines des approches de clustering examinées jusqu'à présent, dont celle de (Olteanu *et al.*, 2013), admettent qu'une utilisation déraisonnable de toutes les informations du graphe (les vecteurs d'attributs des nœuds et leurs liens) peut conduire à identifier de mauvais clusters.

Dans la littérature, on retrouve souvent le terme malédiction de la dimension. Ce dernier fait référence aux problèmes qui apparaissent lors de l'analyse de données de grande dimension, qui ne se produisent pas dans des espaces de faible dimension, en particulier les problèmes de la « rareté », et de la « proximité » des données. Ce phénomène motive le développement des approches de clustering axées sur l'identification des attributs discriminants, pour produire des clusters bien séparés. Cette approche générale est connue sous le nom de clustering de sous-espaces (Subspace clustering). Les méthodes de clustering de sous-espaces sont conçues pour sélectionner les sous-ensembles de dimensions les plus représentatifs. Ils recherchent les projections des données dans différentes dimensions, et identifient des clusters pertinents localement pour certains de ces sous-espaces. Cette technique a également été appliquée au cas des graphes attribués. Cependant, trouver des projections pertinentes est une tâche non triviale. Le choix final des clusters à conserver est également coûteux, et nécessite une étape d'optimisa-

tion combinant la meilleure taille, densité, entropie, dimensionnalité et toute autre fonction de qualité pertinente. De plus, comme chaque cluster est pertinent dans son propre sous-espace, cela a pour effet de produire des clusters superposés, et nécessite des efforts supplémentaires pour contrôler le taux de redondance entre les clusters.

Une approche semi-automatisée pour identifier des sous-ensembles d'attributs pertinents a été présentée dans (Cruz *et al.*, 2011), où les auteurs proposent à des annotateurs de sélectionner manuellement leur perspective compositionnelle préférée. Le choix du sous-ensemble d'attributs est donné explicitement en entrée d'un processus de clustering automatique. Différemment, (Günnemann *et al.*, 2013) proposent une méthode entièrement automatisée pour combiner efficacement des clusters de sous-espace, et de sous-graphes. En particulier, ils utilisent leur ancienne méthode GAMer (Günnemann *et al.*, 2014), pour extraire une liste exhaustive de clusters candidats. Cette dernière sélectionne de petits clusters qui optimisent localement une mesure de qualité. Enfin, une méthode de restitution des clusters finaux est appliquée.

La complexité temporelle des approches de clustering des sous-espaces est élevée (d'ordre $\mathcal{O}(n^3)$, avec n qui représente le nombre d'entrées de l'algorithme), mais la découverte de sous-graphes denses dans des sous-espaces sélectionnés s'avère pertinente pour former les différents clusters. Cependant, le nombre élevé de paramètres d'entrée requis (la taille minimale du cluster, dimensionnalité, densité et redondance) peut avoir un impact négatif sur l'application de ces méthodes, car la sélection des valeurs des paramètres appropriées n'est pas une tâche simple. Une solution possible peut être une approche par essai et erreur (Lee et Shim, 2015), qui effectue à plusieurs reprises des tâches de clustering avec différentes combinaisons des valeurs des paramètres, puis sélectionne enfin le résultat le plus satisfaisant.

Les limites des méthodes citées précédemment (Cruz *et al.*, 2011; Günnemann *et al.*, 2014; Günnemann *et al.*, 2013) sont que : (1) elles ne capturent qu'une partie des informations du graphe (les relations superficielles entre le contenu et les données de structure), et (2) elles sont directement appliquées sur les graphes originaux (en entrée), où le nombre de liens effectif est bien inférieur au nombre de liens possibles (graphe complètement connecté). Par conséquent, ces méthodes ne peuvent pas exploiter efficacement la structure du graphe, ou l'interaction entre la structure du graphe et les informations sur les attributs du nœud.

2.2 Approches modernes du clustering profond des graphes attribués

Nous allons aborder dans cette section deux revues de littérature, la première concerne le développement du clustering profond en général. Ensuite, nous présenterons les approches liées uniquement au clustering des graphes attribués.

2.2.1 Développement du clustering profond

Durant les dernières années, de multiples avancés ont été réalisés dans le domaine du clustering des images, textes et structures graphiques grâce notamment aux autoencodeurs. Un autoencodeur standard (AE) est un réseau de neurones qui permet de compresser les données en entrée en un espace latent, puis reconstruit les données initiales à partir de la représentation compacte. L'objectif principal est de proposer une version reconstruite des données, de manière à ce qu'elle soit la plus semblable possible aux données en entrée. Un autoenodeur standard (AE) est constitué de trois blocs principaux, un encodeur, un module de représentation caché et un décodeur. L'encodeur transforme les données originales en utilisant des filtres, pour apprendre des descripteurs pertinents. Ces derniers seront projetés ensuite dans l'espace latent. Le décodeur, quant à lui, génère une reconstruction

des données à partir des descripteurs compressés. Les autoencodeurs apprennent les caractéristiques les plus pertinentes, permettant de discriminer les données initiales d'une tâche donnée. Ceci est dû aux transformations non linéaires, et aux représentations apprises à chaque sortie de couche, dans les modules d'encodeur et de décodeur. Enfin, ces derniers peuvent utiliser des couches préentraînées d'un autre modèle, afin d'appliquer un apprentissage par transfert.

La figure 2.1 ci-dessous décrit l'architecture de l'autoencodeur.

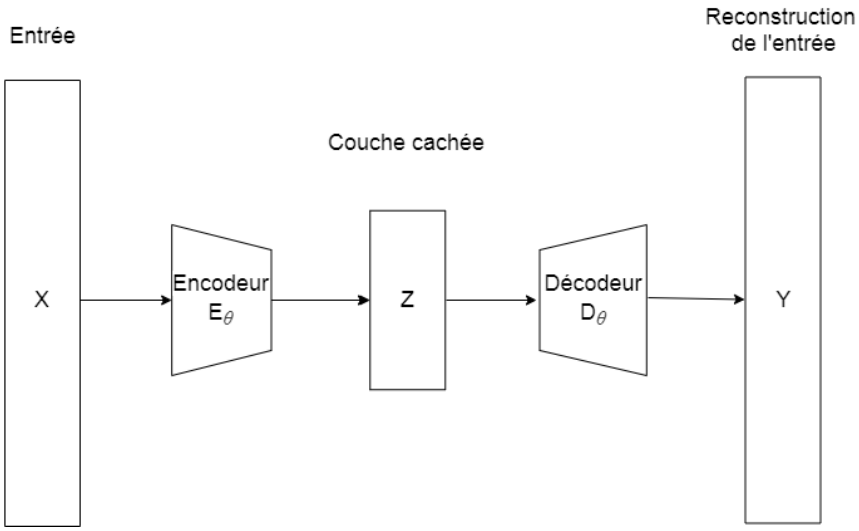


Figure 2.1: Architecture d'un Autoencodeur (AE).

L'autoencodeur variationnel (VAE) (Kingma et Welling, 2014), est considéré comme une variante générative de AE (c.-à-d., capables de générer de nouvelles instances qui semblent provenir du jeu d'entraînement). Le VAE applique la même représentation latente de l'autoencodeur standard (AE), en la rendant plus prévisible, plus continue, et plus dense. Celle-ci suit une distribution prédéfinie (généralement la distribution gaussienne). En forçant les caractéristiques de l'espace de représentation à suivre une distribution gaussienne, le VAE optimise la fonction de coût de l'inférence variationnelle via la descente du gradient stochastique (SGD), et la rétropropagation standard. Il effectue ensuite le reparamétrage de la borne

inférieure variationnelle (astuce de reparamétrage), pour produire un estimateur bayésien du gradient stochastique. Cet estimateur est utilisé pour une reconstruction efficace. L'autoencodeur variationnel tente de minimiser deux éléments : (1) la différence entre l'entrée de l'encodeur et la sortie du décodeur, et (2) la différence entre la distribution de l'encodeur et la distribution de probabilité des données.

La fonction de coût de l'autoencodeur variationnel peut être formulée comme suit :

$$\mathcal{L}(\theta, \phi; X) = \sum_{i}^{N} \left(-D_{KL} \left(q_\phi \left(z \mid x^{(i)} \right) \| p(z) \right) + \mathbb{E}_{q_\phi \left(z \mid x^{(i)} \right)} \left[\log p_\theta \left(x^{(i)} \mid z \right) \right] \right) \tag{2.1}$$

où $X = \left\{ x^{(i)} \right\}_{i=1}^{N}$ représente l'ensemble de données, $p(z)$ est la distribution a priori sur les variables latentes, $q_\phi \left(z \mid x^{(i)} \right)$ est l'approximation variationnelle des données à compresser, $p_\theta \left(x^{(i)} \mid z \right)$ relate la fonction de vraisemblance, et D_{KL} représente la divergence de KL (Kullback-Leibler).

Compte tenu des avantages présentés par les autoencodeurs, d'autres modèles sont apparus dans le domaine du clustering profond. Parmi les approches largement citées dans ce domaine, on retrouve le Deep Clustering Network (DCN) (Yang *et al.*, 2017) qui fut le premier modèle combinant les autoencodeurs et l'algorithme K-Means pour effectuer la tâche du clustering profond. En effet, le DCN alterne entre la reconstruction des données originales, et la formation des clusters, en appliquant une fonction objectif conjointe. Celle-ci combine la fonction de coût de la reconstruction (autoencodeur), et la fonction de coût du clustering (K-Means). Le DCN utilise un algorithme d'optimisation alternatif qui propose une initialisation du clustering via une étape de préentraînement. Ensuite, pendant l'entraînement du modèle, une mise à jour régulière des paramètres du clustering (les centres de clusters et le nombre d'objets attribué à chaque cluster) est effectuée. Ainsi, la

fonction de coût de la reconstruction est définie comme suit :

$$\mathcal{L}_{\text{rec}} = \ell\left(\boldsymbol{g}\left(\boldsymbol{f}\left(\boldsymbol{x}_i\right)\right), \boldsymbol{x}_i\right) \tag{2.2}$$

où ℓ représente la mesure de la reconstruction des données (la méthode des moindres carrés avec $\ell(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2^2$). $\boldsymbol{f}$ désigne la fonction de transformation de la donnée initiale $\boldsymbol{x}_i$ de l'encodeur, et $\boldsymbol{g}$ désigne la fonction de reconstruction de la donnée transformée $\boldsymbol{f}(\boldsymbol{x}_i)$.

La fonction de coût du clustering est définie comme suit :

$$\mathcal{L}_{\text{clus}} = \|\boldsymbol{f}\left(\boldsymbol{x}_i\right) - \boldsymbol{M}\boldsymbol{s}_i\|_2^2 \tag{2.3}$$

où s_i est le vecteur d'affectation de la donnée $\mathbf{x_i}$, et $\boldsymbol{M}\boldsymbol{s}_i$ correspond aux centroïdes des clusters contenus dans le vecteur d'affectation s_i.

Enfin, la fonction objectif du DCN est définie comme suit :

$$\mathcal{L}_{\text{DCN}} = \mathcal{L}_{\text{rec}} + \frac{\lambda}{2}\mathcal{L}_{\text{clus}} \tag{2.4}$$

où $\lambda \geq 0$ représente le paramètre d'équilibre entre la fonction de coût de reconstruction et celle du clustering.

La figure 2.2 de la page suivante décrit l'architecture du DCN.

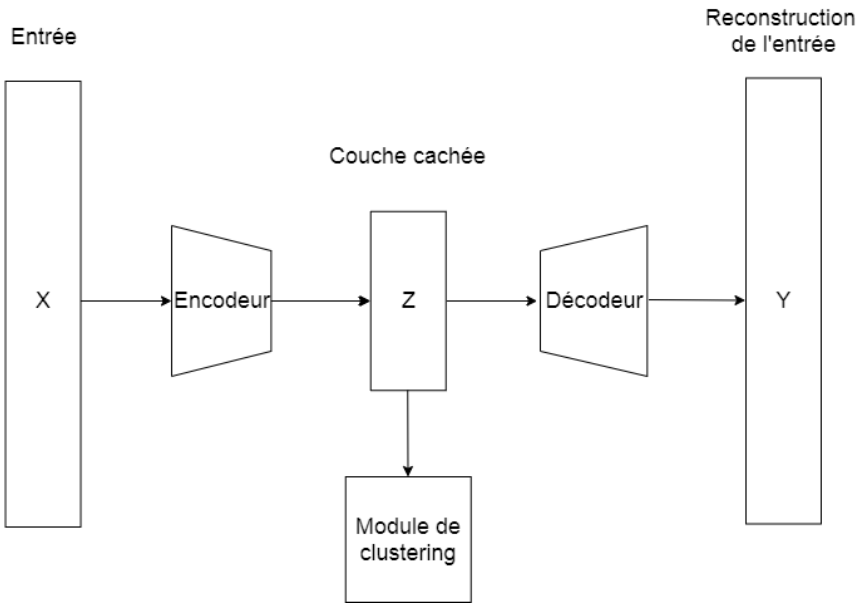


Figure 2.2: Architecture du Deep clustering network (DCN).

Le Deep embedding network (DEN) (Huang *et al.*, 2014) est une approche de clustering profond qui se base sur un autoencodeur standard. DEN utilise l'encodeur pour transformer les données originales, puis le décodeur reconstruit les données en minimisant la fonction de coût de reconstruction. En prenant en compte les descripteurs appris lors de la phase d'encodage, deux contraintes sont alors appliquées pour rendre l'apprentissage orienté clustering. La première est une contrainte de persistance de la localité, elle vise à mieux représenter les informations locales de chaque donnée originale dans l'espace latent. La deuxième contrainte concerne l'amélioration de la répartition des points latents de chaque cluster, elle vise à diagonaliser les représentations apprises par blocs, et les regrouper, afin d'améliorer la formation des clusters. La contrainte de localité, et celle de la répartition des ensembles de représentation des clusters constituent la fonction de coût du clustering. La fonction de coût de la reconstruction, combiné à celle du clustering, forment la fonction objectif du DEN. Enfin, après l'apprentissage des représentations, l'algorithme K-Means est appliqué pour former les clusters finaux.

La fonction de coût de la reconstruction du DEN est définie comme suit :

$$\mathcal{L}_{\text{rec}} = -\sum_{i=1}^{N} \left[\mathbf{x}_i \log \hat{f}\left(\mathbf{x}_i\right) + \left(1 - \mathbf{x}_i\right) \log \left(1 - \hat{f}\left(\mathbf{x}_i\right)\right) \right] \tag{2.5}$$

où $\hat{f}(\mathbf{x})$ représente la fonction de reconstruction du décodeur $\hat{f}$, et de la donnée originale $\mathbf{x_i}$.

La fonction de coût de la contrainte de localité du DEN est formulée comme suit :

$$\mathcal{L}_1 = \sum_{i,j \in d(i)} S_{ij} \left\| f\left(\mathbf{x}_i\right) - f\left(\mathbf{x}_j\right) \right\|^2 \tag{2.6}$$

où $f(\mathbf{x})$ représente la fonction de transformation de l'encodeur par rapport aux données originales $\mathbf{x_i}$ et $\mathbf{x_j}$, S_{ij} désigne la fonction de similarité ($S_{ij} = e^{-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{t}}$, où t est un paramètre de réglage) entre $\mathbf{x_i}$ et $\mathbf{x_j}$, et $d(i)$ est l'ensemble contenant les indices des d plus proches voisins du point i.

La fonction de coût de la répartition des clusters est définie comme suit :

$$\mathcal{L}_2 = \sum_{i=1}^{N} \sum_{k=1}^{K} \lambda_k \left\| f^k\left(\mathbf{x}_i\right) \right\| \tag{2.7}$$

où λ_k représente le poids du cluster k, et f^k désigne la fonction de transformation de l'encodeur $f\left(\mathbf{x}_i\right)$ du cluster k $\left\{ f^k\left(\mathbf{x}_i\right) \right\}_{k=1}^{K}$.

Ainsi, la fonction de coût combiné, de la reconstruction et du clustering, est définie comme suit :

$$\mathcal{L}_{\text{DEN}} = \mathcal{L}_{rec} + \alpha\mathcal{L}_1 + \beta\mathcal{L}_2 \tag{2.8}$$

avec α et β des paramètres de pondération, qui admettent des valeurs entre 0 et 1.

Le Deep Embedded Clustering (DEC) (Xie *et al.*, 2016) est l'une des méthodes les plus représentatives du clustering profond. L'apprentissage du DEC utilise un autoencodeur standard et se compose de deux phases. Dans la première étape, un préentraînement est effectué pour initialiser les paramètres de l'encodeur et du décodeur, en appliquant uniquement la fonction de coût de la reconstruction de l'autoencodeur. Puis, dans la deuxième étape, le décodeur est retiré, et l'encodeur est affiné en optimisant la divergence de KL (Kullback-Leibler) entre q_{ik} (la probabilité d'affectation de l'objet i au cluster k) et p_{ik} (la fréquence d'affectation de l'objet i par rapport au cluster k). En utilisant la rétropropagation standard, et la descente de gradient stochastique (SGD), l'apprentissage de l'encodeur vise à produire des points latents pour minimiser la fonction de coût. Ce processus est répété jusqu'à ce que les affectations de cluster soient stables.

La probabilité d'affectation de l'objet i au cluster k est définie comme suit :

$$q_{ik} = \frac{(1 + ||z_i - \mu_k||^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{k'}(1 + ||z_i - \mu_{k'}||^2/\alpha)^{-\frac{\alpha+1}{2}}} \tag{2.9}$$

où $z_i = f_\theta(x_i) \in Z$ correspondent à $x_i \in X$ après l'encodage.

Dans l'article (Xie *et al.*, 2016), les auteurs fixent $\alpha = 1$, donc l'équation (2.9) peut être simplifiée en :

$$q_{ik} = \frac{(1 + ||z_i - \mu_k||^2)^{-1}}{\sum_{k'}(1 + ||z_i - \mu_{k'}||^2)^{-1}} \tag{2.10}$$

La fréquence d'affectation p_{ik} des objets i par rapport aux clusters k est définie comme suit :

$$p_{ik} = \frac{q_{ik}^2 / f_k}{\sum_{k'} q_{ik'}^2 / f_{k'}} \tag{2.11}$$

où $f_k = \sum_i q_{ik}$ représente la fréquence des affectations de l'objet i au cluster k.

Enfin, la fonction de coût à minimiser est définie comme la divergence de KL (Kullback-Leibler) entre q_{ik}, et p_{ik} :

$$\mathcal{L}_{DEC} = KL(P||Q) = \sum_i \sum_k p_{ik} \log \frac{p_{ik}}{q_{ik}} \tag{2.12}$$

Le Variational Deep Embedding (VaDE) (Jiang *et al.*, 2017), combine l'utilisation de l'autoencodeur variationnel (VAE) et des mélanges de modèles gaussiens (Gaussian Mixture Model : GMM). En premier lieu, VaDE encode les données en entrée, puis utilise les GMM pour initialiser les composantes des clusters. Ensuite, le décodeur se charge de la reconstruction des données. Simultanément, le modèle tend à minimiser la fonction de coût de la reconstruction, et la divergence de KL (la fonction de coût du clustering), afin d'optimiser la fonction de coût du modèle (ELBO [4]). Contrairement au VAE, qui utilise un a priori gaussien unique, où les

4. ELBO (Evidence Lower Bound) est une fonction de coût utilisée dans l'inférence variationnelle.

représentations latentes sont indépendantes et identiquement distribuées, VaDE utilise des mélanges de modèles gaussiens, où les covariances entre les données sont prises en compte, ce qui le rend plus adapté aux tâches de clustering. La figure 2.3 ci-dessous décrit l'architecture du VaDE.

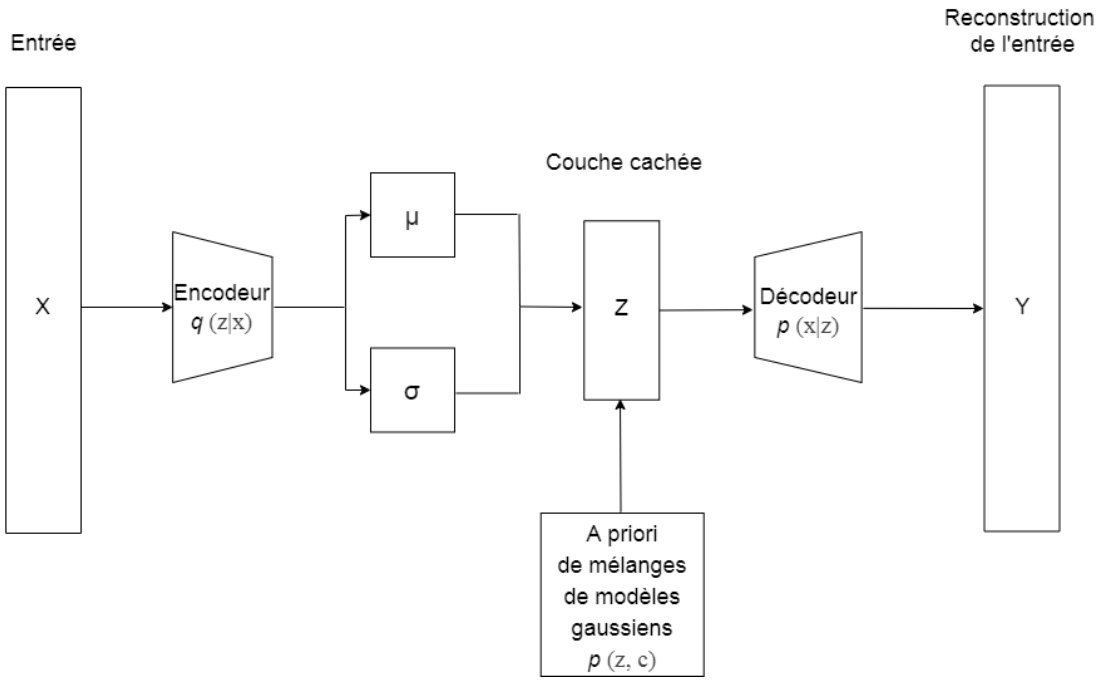


Figure 2.3: Architecture du Variational Deep Embedding (VaDE).

La fonction de coût de VaDE (ELBO) est définie comme suit :

$$\mathcal{L}_{ELBO}(\boldsymbol{x}) = E_{q(z,c|\boldsymbol{x})}[\log p(\boldsymbol{x} \mid z)] - D_{KL}(q(z,c \mid x) \| p(z,c)) \tag{2.13}$$

où $p(\boldsymbol{x} \mid z)$ désigne la distribution de Bernoulli, $q(z,c \mid \boldsymbol{x})$ signifie le postérieur variationnel, et $p(z,c)$ représente le a priori des mélanges de modèles gaussiens.

Le premier terme de l'équation (2.13) représente la fonction de coût de reconstruction, et le second terme désigne la divergence de KL (Kullback-Leibler) par rapport au mélange de modèles gaussiens. Dans ce cas, le second terme est consi-

déré comme la fonction de coût du clustering.

En résumé, nous avons illustré dans cette partie une description globale des méthodes de clustering profond. En ce qui concerne la fonction de coût, les approches présentées optimisent conjointement la fonction objectif du clustering et la fonction objectif globale du modèle ; cependant, la complexité de calcul des méthodes présentées varie beaucoup. Pour les algorithmes basés sur un autoencodeur standard, la complexité de calcul est fortement liée à la fonction de coût du clustering. Enfin, les modèles basés sur les autoencodeurs variationnels ont généralement une complexité de calcul plus élevée que les méthodes basées sur les autoencodeurs standards, notamment l'algorithme DEC (DEC dispose d'une complexité linéaire par rapport aux données, elle lui permet de s'adapter à de grands ensembles de données).

2.2.2 Revue des méthodes de clustering profond pour les graphes attribués

Nous nous intéresserons dans cette section aux approches basées sur le clustering profond pour les graphes attribués. Premièrement, une revue des méthodes de clustering profond utilisant des autoencodeurs de débruitage est présentée. Ensuite, nous allons aborder les approches utilisant les autoencodeurs de graphes.

Le Deep Neural Network for Graph Representations (DNGR) (Cao *et al.*, 2016), utilise un autoencodeur de débruitage empilé (Stacked Denoising Autoencoder : SDAE) pour encoder la matrice d'adjacence du graphe, via des perceptrons multicouches, afin d'apprendre des réseaux d'intégrations (*embedding*[5]) de chaque nœud, et préserver les proximités d'ordre élevées (les nœuds du voisinage immédiat et non immédiat). Le processus d'apprentissage se déroule en trois étapes. La

5. Un *embedding* dans le contexte des graphes est une représentation vectorielle de faible dimension d'un graphe, qui préserve les informations topologiques d'un nœud.

première étape capture les informations structurelles (liens entre les nœuds) du graphe et génère une matrice de co-occurrence probabiliste. La deuxième étape calcule la matrice PPMI [6] (Positive Pointwise Mutual Information) à partir de la matrice de cooccurrence probabiliste générée dans la phase précédente. Puis, SDAE est utilisée dans la troisième étape, pour extraire la représentation latente des nœuds du graphe. Il est à noter que DNGR ignore les vecteurs des caractéristiques des nœuds, ils ne considèrent que les informations structurelles des graphes (liens entre les nœuds).

De même, le Structural Deep Network Embedding (SDNE) (Wang *et al.*, 2016) utilise un autoencodeur de débruitage empilé (SDAE), pour encoder la matrice d'adjacence du graphe. SDNE impose deux contraintes pour préserver conjointement la proximité du premier ordre (les nœuds du voisinage immédiat), et la proximité du second ordre (les nœuds du voisinage non immédiat) entre les nœuds. La fonction de coût de l'encodeur permet aux réseaux d'intégrations (*embedding*) appris de préserver la proximité du premier ordre du nœud, en minimisant la distance entre le réseau d'intégration d'un nœud et ceux de ses voisins. Comme le DNGR, SDNE ignore les vecteurs des attributs des nœuds, ils ne considèrent que les informations topologiques des graphes.

Ainsi, la première fonction de coût $\mathcal{L}_{c1}$ du SDNE est définie comme suit :

$$\mathcal{L}_{c1} = \sum_{i,j=1}^{n} s_{i,j} \left\| \mathbf{y}_i - \mathbf{y}_j \right\|_2^2 \tag{2.14}$$

où $s_{i,j}$ représente le lien reliant le nœud i au nœud j du graphe en entrée. $\mathbf{y}_i$, et

6. Positive Pointwise Mutual Information est une mesure quantifiée de la probabilité que deux événements se produisent, compte tenu de leurs probabilités individuelles, et par rapport au cas où les deux événements sont complètement indépendants (Niwa et Nitta, 1994).

$\mathbf{y}_j$ désignent la représentation latente des nœuds i et j.

La deuxième fonction de coût permet aux réseaux d'intégrations appris de préserver la proximité de second ordre du nœud, en minimisant la distance entre les entrées originales d'un nœud et ses entrées reconstruites. Ainsi, la deuxième fonction de coût $\mathcal{L}_{c2}$ est définie comme suit :

$$\mathcal{L}_{c2} = \sum_{i=1}^{n} \|(\hat{\mathbf{x}}_i - \mathbf{x}_i) \odot \mathbf{b_i}\|_2^2 \tag{2.15}$$

où $\mathbf{x}_i$ représente le nœud en entrée i, $\hat{\mathbf{x}}_i$ désigne la sortie reconstruite du nœud i et $\mathbf{b_i}$ désigne le biais d'apprentissage du perceptron multicouche. Le signe $\odot$ symbolise le produit commutatif et associatif en algèbre linéaire (Suetin *et al.*, 1997).

Afin d'assurer la préservation des contraintes citées précédemment, la fonction de coût globale du SDNE est définie comme suit :

$$\mathcal{L}_{SDNE} = \alpha\mathcal{L}_{c1} + \mathcal{L}_{c2} + \beta\mathcal{L}_{reg} \tag{2.16}$$

où $\mathcal{L}_{reg}$ est un terme de régularisation de la norme L2 [7] pour éviter le surapprentissage. α et β sont des paramètres de pondération, qui admettent des valeurs dans entre 0 et 1.

Les autoencodeurs de graphes (GAE : Graph Autoencoders) (Kipf et Welling, 2016) sont des réseaux de neurones profonds qui compressent la structure du graphe en entrée (les liens ainsi que les vecteurs d'attributs des nœuds) dans un

7. La norme L2 mesure la distance la plus courte depuis l'origine. Elle est définie comme la racine de la somme des carrés des composantes d'un vecteur dans l'espace euclidien (Li, 2009).

espace de représentation latent, puis décodent les informations des descripteurs en un graphe reconstruit. Les GAE peuvent être utilisés pour apprendre les réseaux d'intégration (*embeddings*). De ce fait, l'autoencodeur de graphe exploite les réseaux de neurones convolutifs de graphes (GCN : Graph Convolutional Networks) (Kipf et Welling, 2017) pour encoder les informations structurelles du nœud, et les vecteurs d'attributs du nœud en même temps. Le décodeur du GAE vise à décoder les informations relationnelles des nœuds à partir de leurs réseaux d'intégrations (*embeddings*), en reconstruisant la matrice d'adjacence du graphe. Ainsi, GAE est entraîné en optimisant une fonction objective (par exemple : en minimisant l'entropie croisée négative) qui évalue la correspondance entre la matrice d'adjacence réelle $\mathbf{A}$ et la matrice d'adjacence reconstruite $\hat{\mathbf{A}}$.

L'autoencodeur variationnel de graphe (VGAE : Variational Graph Autoencoder) (Kipf et Welling, 2016) est une version variationnelle de GAE, qui apprend une distribution des données. VGAE suppose que la distribution empirique $q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A})$ doit être aussi proche que possible de la distribution a priori $p(\mathbf{Z})$.

Pour cela, VGAE (Kipf et Welling, 2016) optimise la borne inférieure variationnelle $\mathcal{L}$ comme suit :

$$\mathcal{L} = E_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{A} \mid \mathbf{Z})] - KL[q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}) \| p(\mathbf{Z})] \tag{2.17}$$

où $KL(\cdot)$ désigne la fonction de divergence de Kullback-Leibler qui mesure la distance entre deux distributions, et $p(\mathbf{Z})$ indique une variable a priori gaussienne.

Le Marginalized Graph Autoencoder for Graph Clustering (MGAE) (Wang *et al.*, 2017a) exploite un autoencodeur de graphe pour reconstruire la matrice d'adjacence. Le MGAE introduit un bruit aléatoire dans les vecteurs d'attributs des nœuds, afin de rendre l'apprentissage plus robuste aux variations. Puis, le lapla-

cien est calculé et affiné selon la matrice d'adjacence pour appliquer un clusterting spectral. La séparation des tâches de clustering et de reconstruction rend le MGAE moins efficace que les modèles plus récents.

Adversarially Regularized Graph Auto-Encoder (ARGA) (Pan *et al.*, 2018) et Adversarially Regularized Variational Graph Auto-Encoder (ARVGA) (Pan *et al.*, 2018) sont deux modèles de clustering de graphes avec la même architecture et des fonctions de coût similaires. ARGA utilise un autoencodeur de graphe (GAE) pour reconstruire les représentations latentes. ARVGA, quant à lui, adopte l'autoencodeur de graphes variationnel (VGAE), celui-ci applique l'inférence variationnelle pour contraindre les représentations latentes à être proches de zéro. ARGA et ARVGA, exploitent aussi une régularisation contradictoire (Adversarial Regularization), celle-ci agit telle qu'un réseau discriminateur (DN : Discriminator Network), pour forcer les représentations latentes à correspondre à une distribution antérieure. Néanmoins, ARGA et ARVGA héritent des limitations des Réseaux antagonistes génératifs (GAN : Generative Adversarial Networks) (Goodfellow *et al.*, 2020) telles que l'effondrement de mode [8] (Mode Collapse) et l'échec de la convergence.

Deep Attentional Embedding for Graph Clustering (DAEGC) (Wang *et al.*, 2019) est un modèle de clustering de graphes attribués basé sur un autoencodeur de graphe. La fonction de coût de clustering de DAEGC est similaire à celle du DEC (Xie *et al.*, 2016). La contribution majeure du DAEGC réside dans l'utilisation des mécanismes d'attention dans les réseaux de neurones convolutifs de graphes (GCN), qui permet de mieux ajuster la représentation des voisins pour chaque nœud dans la représentation latente. Néanmoins, l'utilisation des méca-

8. Dans le contexte des GAN, un effondrement de mode signifie que le générateur de données produit un seul type de sortie, ou un petit ensemble de sorties, de telle manière à ce que ses sorties piègent éternellement le discriminateur.

nismes d'attention dans les GCN augmente le temps d'apprentissage du DAEGC, ce qui le rend plus lent par rapport aux autres modèles (GAE, VGAE, ARGA, et ARVGA).

GMM-VGAE (Hui *et al.*, 2020) est un modèle de clustering profond des graphes attribués basés sur un autoencodeur de graphe. La fonction de coût du clustering est une combinaison de la fonction de coût de reconstruction d'un autoencodeur de graphe, avec une fonction de coût de clustering KL similaire à celle du VaDE (Jiang *et al.*, 2017). La contribution majeure du GMM-VGAE est l'utilisation des mélanges de modèles gaussiens, afin de découvrir les distributions de données complexes. En termes de performance, le GMM-VGAE dépasse les modèles cités précédemment dans cette section.

2.3 Revue des méthodes d'apprentissage dynamique

Les modèles d'apprentissage dynamiques constituent un sujet de recherche qui suscite davantage d'intérêt en apprentissage profond. Par rapport aux modèles statiques qui sont régis par une architecture et un paramétrage statiques, les réseaux dynamiques peuvent adapter leurs structures, ou leurs paramètres à différents niveaux, conduisant à des avantages notables en termes de précision, d'efficacité, de calcul et d'adaptabilité. Plusieurs variantes de modèles dynamiques sont disponibles, on y retrouve principalement des modèles par instance (Huang *et al.*, 2017; Teerapittayanon *et al.*, 2016; Bengio *et al.*, 2015) qui traitent chaque instance avec des architectures dynamiques, ou le paramétrage dynamique (Yang *et al.*, 2019; Cheng *et al.*, 2020; Gao *et al.*, 2020) qui introduit des paramètres dépendants des données.

2.3.1 Architectrures dynamiques

Étant donné que différentes tâches et problématiques à résoudre dans l'apprentissage profond peuvent avoir des exigences de calcul diverses, il est naturel d'effectuer un apprentissage avec des architectures dynamiques conditionnées sur chaque échantillon de données. Plus précisément, on peut ajuster la profondeur du réseau, ou sa largeur.

Les réseaux dotés d'architectures dynamiques permettent non seulement d'économiser des calculs redondants pour les instances canoniques « simples à apprendre et à distinguer », mais préservent également leur pouvoir de représentation lors de la reconnaissance d'échantillons non canoniques « durs à apprendre et à distinguer ». Une telle propriété conduit à des avantages remarquables en termes d'efficacité par rapport aux techniques d'accélération des modèles statiques. Ces derniers gèrent des entrées « faciles » (les entités de données et les relations sont présentées sous la forme la plus simple possible), et « dures » (les entités de données et les relations admettent des variantes) avec un calcul identique, mais ne parviennent pas à réduire la redondance intrinsèque de calcul.

Alors que les modèles d'apprentissage profond modernes deviennent de plus en plus profonds, leur objectif est de reconnaître des échantillons plus « durs à apprendre ». Une solution simple pour réduire les calculs redondants consiste à effectuer un apprentissage des modèles avec des profondeurs dynamiques. On retrouve parmi ces méthodes :

1. La sortie précoce : cette méthode permet d'apprendre des échantillons « faciles », avec des sorties peu profondes, sans exécuter de couches plus profondes (Huang *et al.*, 2017; Izhikevich, 2003).
2. Le saut de couche : cette approche permet de faire un saut sélectif des

couches des réseaux de neurones intermédiaires conditionnées à chaque instance (Graves, 2016; Teerapittayanon *et al.*, 2016; Bolukbasi *et al.*, 2017). En raison de l'exécution séquentielle par couche dans les réseaux de neurones profonds, les modèles avec des profondeurs dynamiques bénéficient d'une efficacité d'exécution élevée.

3. La largeur dynamique des couches entièrement connectées (FC : Fully connected) : cette approche se base sur le principe de l'activation des unités neuronales, en fonction de la tâche d'apprentissage. En effet, les unités neuronales sont responsables de la représentation des différentes caractéristiques latentes des données, mais leur activation n'est pas impérative pour chaque instance. Les principaux travaux réalisés avec cette propriété apprennent à contrôler de manière adaptative les activations neuronales par des unités auxiliaires (Shen *et al.*, 2020; Wu *et al.*, 2018; Bengio, 2013). Bien que chaque couche du réseau de neurones soit toujours exécutée, les unités neuronales sont activées sélectivement en fonction de l'instance apprise.
4. Le mélange d'experts (MoE : Mixture-of-Experts) : cette approche permet d'améliorer la capacité des réseaux de neurones, sans les approfondir. L'ensemble de modèles experts est obtenu en cascade, de manière conditionnelle, et sur la base des premières étapes d'apprentissage (Bengio *et al.*, 2015). En effet, pendant l'apprentissage, plusieurs branches de réseaux de neurones sont sélectionnées, afin d'examiner des parties différentes de l'ensemble de données. Ces experts pourraient être exécutés de manière sélective, et leurs sorties sont fusionnées avec des poids dépendant des données.

2.3.2 Paramétrage dynamique

Bien que les architectures dynamiques peuvent adapter leurs graphes d'inférence à chaque instance, et obtenir une allocation efficace des calculs, ils ont généralement

des conceptions d'architecture spéciales, nécessitant des stratégies d'apprentissage spécifiques, ou un réglage minutieux des paramètres. En général, l'adaptation des paramètres peut être réalisée à partir de trois aspects :

1. l'ajustement des paramètres appris sur la base de l'entrée : cette approche (Dai *et al.*, 2017; Zhu *et al.*, 2019; Gao *et al.*, 2020) permet l'adaptation des paramètres. Elle consiste à ajuster les poids en fonction de leur entrée lors de l'apprentissage.
2. Génération directe des paramètres du réseau de neurones à partir de l'entrée : cette technique consomme généralement peu de calculs pour obtenir les ajustements (Cheng *et al.*, 2020).
3. Redimensionnement des fonctionnalités avec une attention *soft*[9] (Soft Attention) : cette approche utilise les poids d'attention (Yang *et al.*, 2019), pour mettre en évidence les caractéristiques pertinentes.

Néanmoins, peu de recherches ont été effectuées pour analyser les modèles d'apprentissage dynamiques d'un point de vue théorique. Nous énumérons ci-dessous, deux problèmes théoriques fondamentaux :

1. La prise de décision optimale dans les réseaux dynamiques : c'est une opération essentielle dans la plupart des réseaux dynamiques (en particulier ceux conçus pour améliorer l'efficacité de calcul). Elle permet de prendre des décisions dépendantes des données, par exemple, déterminer si un module doit être évalué ou ignoré. Les solutions existantes utilisent soit des critères basés sur la confiance, ou des fonctions de contrôle.

9. Dans les mécanismes d'attention, une attention *soft* permet d'attribuer des poids élevés pour les caractéristiques pertinentes des entrées de la tâche à résoudre, afin de les mettre en évidence.

2. L'existence de problèmes de généralisation dans les modèles d'apprentissage dynamiques : cette problématique apparait lorsque l'apprentissage est effectué avec de petits sous-réseaux sur des échantillons « faciles », tandis que les sous-réseaux plus grands apprennent sur des échantillons plus « difficiles ». Ceci conduit à une divergence entre la distribution des données d'apprentissage et celle de l'étape d'inférence.

2.4 Caractère arbitraire des descripteurs

Nous désignons par « Caractère arbitraire des descripteurs », les descripteurs appris par un modèle d'apprentissage profond, qui ne possèdent pas les mêmes étiquettes que celles prédites par celui-ci. En effet, il est possible de former un réseau de neurones à l'aide d'étiquettes prédites, et extraites sur la base d'hypothèses. Cependant, ces étiquettes construites sont moins efficaces que les vraies étiquettes, car certaines d'entre elles ne correspondent pas aux catégorisations réelles.

Ainsi, (Mrabah *et al.*, 2020) propose un modèle de clustering profond, basé sur un apprentissage dynamique permettant de réduire graduellement ce phénomène. Ce dernier, intitulé Dynamic Autoencoder (DynAE), admet une fonction de coût dynamique permettant de mieux exploiter les connaissances graduelles et incertaines acquises par la pseudosupervision. En éliminant progressivement la fonction de coût de reconstruction en faveur d'une construction.

DynAE se base sur un autoencodeur, il dispose d'une étape de préentraînement pour initialiser le modèle et apprendre les caractéristiques des données traitées (des images principalement). L'encodeur et le décodeur sont préentraînés pour optimiser la fonction de coût de l'autoencodeur, et initialiser les poids des paramètres d'entraînement du modèle. Après le préentraînement, l'étape de clustering

profond des images est enclenchée, les poids de l'autoencodeur sont affinés en optimisant une fonction de coût dynamique. Pour cela, K-Means est utilisé pour initialiser, et mettre à jour les centres des clusters dans l'espace latent. Semblable aux modèles de clustering profond basé sur les autoencodeurs, la fonction de coût du DynAE comporte deux parties. La première partie constitue la fonction de coût de reconstruction, et la seconde partie, désigne la fonction de coût du clustering. Cependant, la fonction de coût du DynAE admet des paramètres de contrôle des données conflictuelles, et non conflictuelles, à l'égard de chaque cluster.

Ainsi, la fonction de coût de la reconstruction est définie comme suit :

$$\mathcal{L}_{rec} = \sum_{i=1}^{N} \begin{cases} \|x_i - \hat{x}_i\|_2^2 & \text{si } x_i \in \bar{S} \\ \|g\left(\sigma\left(x_i\right)\right) - \hat{x}_i\|_2^2 & \text{sinon} \end{cases} \tag{2.18}$$

où N est le nombre d'objets, x_i représente l'objet i en entrée, $\hat{x}_i$ désigne la sortie du décodeur, $g\left(.\right)$ signifie la fonction de décodage, $\sigma\left(.\right)$ est la fonction qui génère le centroïde avec l'affectation de clustering la plus élevée, et $\bar{S}$ représente l'ensemble des données conflictuelles.

La fonction de coût du clustering est définie comme suit :

$$\mathcal{L}_{clus} = \sum_{x_i \in S} \|f\left(x_i\right) - \sigma\left(x_i\right)\|_2^2 \tag{2.19}$$

où x_i représente l'objet i en entrée, $f\left(.\right)$ désigne la fonction d'encodage, et $\sigma\left(.\right)$ reflète la fonction qui génère le centroïde avec l'affectation de clustering la plus élevée.

Ainsi, la fonction de coût conjointe du DynAE est définie comme la combinaison des fonctions de coût de la reconstruction et du clustering, et cela sans avoir

recours à un coefficient de balance. Celle-ci est formulée comme suit :

$$\mathcal{L}_{DynAE} = \mathcal{L}_{rec} + \mathcal{L}_{clus} \tag{2.20}$$

2.5 Conclusion

Dans ce chapitre, nous avons présenté les principaux travaux relevant du clustering des graphes attribués, en commençant par les méthodes traditionnelles, dont les problèmes ont été largement dépassés et traités grâce aux méthodes de clustering profond des graphes attribués. Le principal point à discuter pour tous ces modèles, c'est l'absence de travaux traitant la problématique du caractère arbitraire des descripteurs pour les graphes attribués. De ce fait, nous allons présenter dans le chapitre suivant une approche qui traite ce cas, en présentant un modèle de clustering profond dynamique pour les graphes attribués.

CHAPITRE III

APPROCHE PROPOSÉE

Ce chapitre présente une nouvelle approche de clustering des graphes attribués. La méthode proposée se compose de deux phases. La première phase vise à identifier des clusters en utilisant un modèle de clustering profond pour les graphes attribués. La deuxième phase consiste à introduire un processus d'apprentissage dynamique pour améliorer la formation des clusters. Celui-ci prend en compte les nœuds conflictuels (les nœuds dont la distance par rapport au centroïde de leur cluster est supérieure à un seuil fixé), et les nœuds non conflictuels (les nœuds dont la distance par rapport au centroïde de leur cluster est inférieure à un seuil fixé) ; dans le but d'introduire une fonction de coût dynamique, et diminuer l'impact du caractère arbitraire des descripteurs. Notre approche cherche à démontrer l'impact de la méthode d'évaluation dynamique, visant à mieux regrouper les nœuds d'un graphe attribué dans le contexte du clustering profond. Dans ce qui suit, nous commençons par introduire quelques notations et définitions. Ensuite, nous présentons la phase de clustering profond des graphes attribués avec une description détaillée. Enfin, nous présenterons la phase de clustering profond dynamique de notre modèle (D-DGC).

3.1 Notations

Avant de décrire la démarche proposée, nous présentons quelques notations liées au sujet traité. Dans le présent travail, nous utilisons des graphes non orientés $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{X})$, où $\boldsymbol{V} = \{v_1, v_2, \ldots, v_n\}$ désigne un ensemble de nœuds avec $|\boldsymbol{V}| = n$, et $\boldsymbol{E} = \{e_{ij}\}$ représente l'ensemble de liens entre les nœuds. La topologie du graphe $\mathcal{G}$ est déterminée par une matrice d'adjacence $\boldsymbol{A} = \{a_{ij}\} \in \mathbb{R}^{n \times n}$, où $a_{i,j} = 1$ si $(v_i, v_j) \in \boldsymbol{E}$, sinon $a_{i,j} = 0$. $\boldsymbol{X}$ désigne la matrice des vecteurs d'attributs des nœuds, où $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \in \mathbb{R}^{n \times d}$, et $x_i \in \mathbb{R}^d$ représente le vecteur d'attributs du nœud v_i.

L'objectif de ce travail et l'identification de clusters disjoints dans un graphe attribué $\mathcal{G}$, où chaque nœud $v \in V$ peut appartenir à un et un seul cluster c_k. Conjointement à cela, ce travail comportera une phase d'apprentissage profond dynamique, celle-ci introduit une fonction de coût dynamique permettant de réduire le caractère arbitraire des descripteurs, et améliorer les performances du clustering profond des graphes attribués.

3.2 Phase de clustering profond du graphe attribué

Dans cette section, il sera question de détailler notre approche, le modèle de clustering profond des graphes attribués. En effet, ce dernier se compose de deux étapes, la première étape dite de préentraînement, elle consiste à initialiser le modèle de clustering profond avec des paramètres d'entraînement (les paramètres entraînables de l'autoencodeur, les centres de clusters et les covariances), et la deuxième utilisée pour le clustering final du graphe attribué (formation des clusters finaux).

3.2.1 Étape de préentraînement

Le préentraînement est une technique qui permet d'améliorer les performances des modèles d'apprentissage profond, en particulier le clustering profond (Erhan *et al.*, 2010). Dans cette démarche, nous avons reconduit cette approche de préentraînement, dans le but de fournir une initialisation à l'étape de clustering. Dans ce contexte, nous définissons l'étape de préentraînement, qui repose sur l'utilisation d'un autoencodeur variationnel de graphe (VGAE)(Kipf et Welling, 2016). Celui-ci se compose d'un encodeur et d'un décodeur. L'encodeur du VGAE est défini comme suit :

$$\begin{aligned} \boldsymbol{Z}^{(1)} &= f_{\text{Relu}}\left(\boldsymbol{X}, \boldsymbol{A} \mid \boldsymbol{W}^{(0)}\right) \\ \mu = \boldsymbol{Z}_{\mu}^{(2)} &= f_{\text{linéaire}}\left(\boldsymbol{Z}^{(1)}, \boldsymbol{A} \mid \boldsymbol{W}_{\mu}^{(1)}\right) \\ \sigma = \boldsymbol{Z}_{\sigma}^{(2)} &= f_{\text{linéaire}}\left(\boldsymbol{Z}^{(1)}, \boldsymbol{A} \mid \boldsymbol{W}_{\sigma}^{(1)}\right) \end{aligned} \tag{3.1}$$

où $\boldsymbol{W}^{(l-1)}$ indique le poids des paramètres entraînables spécifiques à la couche l, $\boldsymbol{Z}^{(l)}$ représente la sortie de la couche convolutive l, et f signifie la fonction de transformation de chaque couche de l'encodeur.

Étant donné la matrice d'adjacence $\boldsymbol{A}$ (matrice contenant tous les liens entre les nœuds du graphe) et la matrice des vecteurs d'attributs des nœuds $\boldsymbol{X}$, une transformation par couche est effectuée par une fonction f d'un réseau de neurones à convolution de graphes (GCN : Graph convolutional network) (Kipf et Welling, 2017). GCN est décrit dans la figure 3.1 de la page suivante.

f est définie comme suit :

$$f_{\phi}\left(\boldsymbol{Z}^{(l)}, \boldsymbol{A} \mid \boldsymbol{W}^{(l)}\right) = \phi\left(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{Z}^{(l)}\boldsymbol{W}^{(l)}\right) \tag{3.2}$$

où $\boldsymbol{Z}^{(l)}$ est la sortie de la couche l. $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_N$, $\tilde{\boldsymbol{D}}_{ii} = \sum_j \tilde{\boldsymbol{A}}_{ij}$ et $\boldsymbol{I}_N$ est une matrice identité de dimension NxN. $\boldsymbol{W}^{(l)}$ désigne les paramètres de poids entraînables spécifiques à la couche l, et ϕ est une fonction d'activation Relu ($\text{Relu}(\cdot) = \max(0, \cdot)$).

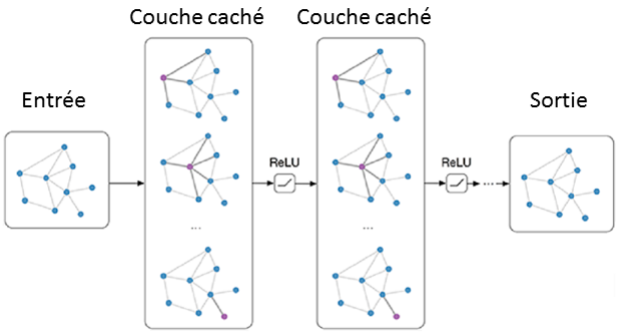


Figure 3.1: Schéma reflétant l'architecture d'un GCN (Kipf et Welling, 2017).

L'architecture de l'encodeur comporte un GCN de deux couches. Formellement, la sortie de l'encodeur est formulée comme suit :

$$
\begin{aligned}
q(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{A}) &= \prod_{i=1}^{n} q\left(z_i \mid \boldsymbol{X}, \boldsymbol{A}\right) \\
q\left(z_i \mid \boldsymbol{X}, \boldsymbol{A}\right) &= \mathcal{N}\left(z_i \mid \boldsymbol{\mu}_i, \operatorname{diag}\left(\boldsymbol{\sigma}_i^2\right)\right)
\end{aligned}
\tag{3.3}
$$

où μ_i désigne le centre associé à chaque nœud v_i, σ_i^2 indique les distances au carré de chaque nœud v_i par rapport à son centre μ_i divisé par le nombre de nœuds n, et $\mathcal{N}$ désigne une distribution normale multidimensionnelle. Il est à rappeler que μ_i, et σ_i^2 sont générés par la deuxième couche de l'encodeur (sortie de l'encodeur).

Ensuite, le calcul de l'ensemble des variables latentes $\boldsymbol{Z}$ est effectué en utilisant une astuce de reparamétrage.

Chaque variable latente z_i est définie comme suit :

$$\begin{aligned} \boldsymbol{z_i} &= \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i * \epsilon \\ \epsilon &\sim \mathcal{N}(0,1) \end{aligned} \tag{3.4}$$

Le décodeur est utilisé, ensuite, pour reconstruire la matrice d'adjacence $\boldsymbol{A}$. Chaque lien e_{ij} de la structure générée par le décodeur est définie par la sigmoïde du produit scalaire entre les variables latentes, z_i du nœud v_i, et z_j du nœud v_j :

$$\begin{aligned} p(\hat{\boldsymbol{A}} \mid \boldsymbol{Z}) &= \prod_{i=1}^{n} \prod_{j=1}^{n} p\left(\hat{\boldsymbol{A}}_{ij} \mid z_i, z_j\right) \\ p\left(\hat{\boldsymbol{A}}_{ij} \mid z_i, z_j\right) &= \phi\left(z_i^{\top}, z_j\right) \end{aligned} \tag{3.5}$$

où $\hat{\boldsymbol{A}}$ représente la matrice d'adjacence reconstruite, ϕ est la fonction d'activation sigmoïde, $\boldsymbol{Z}$ désigne l'ensemble des variables latentes.

Ci-dessous, nous présentons l'architecture globale de l'autoencodeur de graphe utilisé.

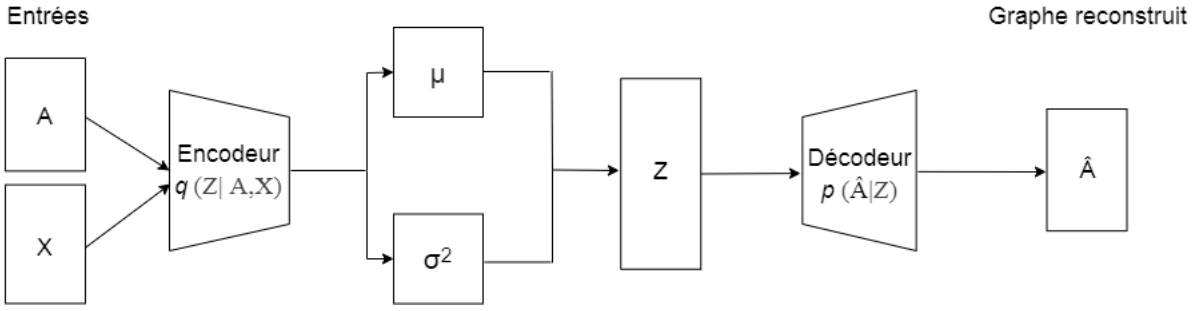


Figure 3.2: Schéma reflétant l'architecture de l'autoencodeur de graphe (Kipf et Welling, 2016).

Le préentraînement du modèle de clustering profond des graphes attribués utilise une fonction de coût composée (reconstruction et régularisation). Afin de les dé-

crire, la fonction de coût de la reconstruction est définie comme l'entropie croisée binaire, permettant de calculer l'erreur de reconstruction entre la matrice d'adjacence du graphe reconstruit $\hat{\boldsymbol{A}}$, et la matrice d'adjacence du graphe en entrée $\boldsymbol{A}$:

$$\mathcal{L}_{rec} = -E_{q(\boldsymbol{Z}|X,A)}[\log p(\hat{A} \mid \boldsymbol{Z})] \tag{3.6}$$

Quant à la fonction de régularisation, elle est formulée de la façon suivante :

$$\mathcal{L}_{reg} = KL[q(\boldsymbol{Z} \mid X, A) \| p(\boldsymbol{Z})] \tag{3.7}$$

où KL signifie la divergence de KL (Kullback-Leibler) entre $q(\boldsymbol{Z} \mid X, A)$ la distribution paramétrée par l'encodeur, et $p(\boldsymbol{Z}) = \mathcal{N}(0, 1)$.

Ainsi, la fonction de coût du préentraînement est le résultat de la combinaison de $\mathcal{L}_{rec}$, qui désigne la fonction de coût de reconstruction, et $\mathcal{L}_{reg}$ qui désigne la fonction de régularisation :

$$\mathcal{L}_{VGAE} = \mathcal{L}_{rec} + \mathcal{L}_{reg} \tag{3.8}$$

Afin d'établir une bonne initialisation des clusters (centres de clusters et covariances), notre approche utilise le mélange de modèles gaussiens (GMM : Gaussian Mixture Model) (Reynolds, 2009), qui par le biais de sa fonction de coût, permet de former des clusters avec des moyennes et des covariances différentes.

Le modèle de mélange gaussien a été appliqué dans le clustering profond (Dilokthanakul *et al.*, 2016) et a donné de bons résultats, ce qui nous a conduits à exploiter cette solution pour le clustering profond des graphes attribués. La théo-

rie du GMM suppose qu'un mélange de distributions gaussiennes est utilisé pour générer les données, celui-ci applique une affectation souple [10] (soft assignement) des nœuds aux clusters. Ainsi, la fonction de coût du GMM est définie comme suit :

$$\mathcal{L}_{GMM} = \log(p(X)) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} p\left(v_i \mid c_k\right) p\left(c_k\right) \right) \tag{3.9}$$

où K représente le nombre de clusters, $p\left(v_i \mid c_k\right)$ désigne la probabilité d'appartenance du nœud v_i au cluster c_k, et $p\left(c_k\right)$ est la probabilité a priori du cluster c_k.

En résumé, dans l'étape de préentraînement, notre approche prend en entrée la matrice d'adjacence du graphe, et les attributs des nœuds, tandis qu'en sortie, elle fournit une matrice d'adjacence reconstruite de la structure du graphe, et une initialisation des clusters et des covariances avec GMM. La figure 3.3 de la page suivante décrit ce processus.

10. Une affectation souple dans le domaine du clustering des graphes attribués désigne l'attribution d'une probabilité d'appartenance de chaque nœud du graphe à chaque cluster formé.

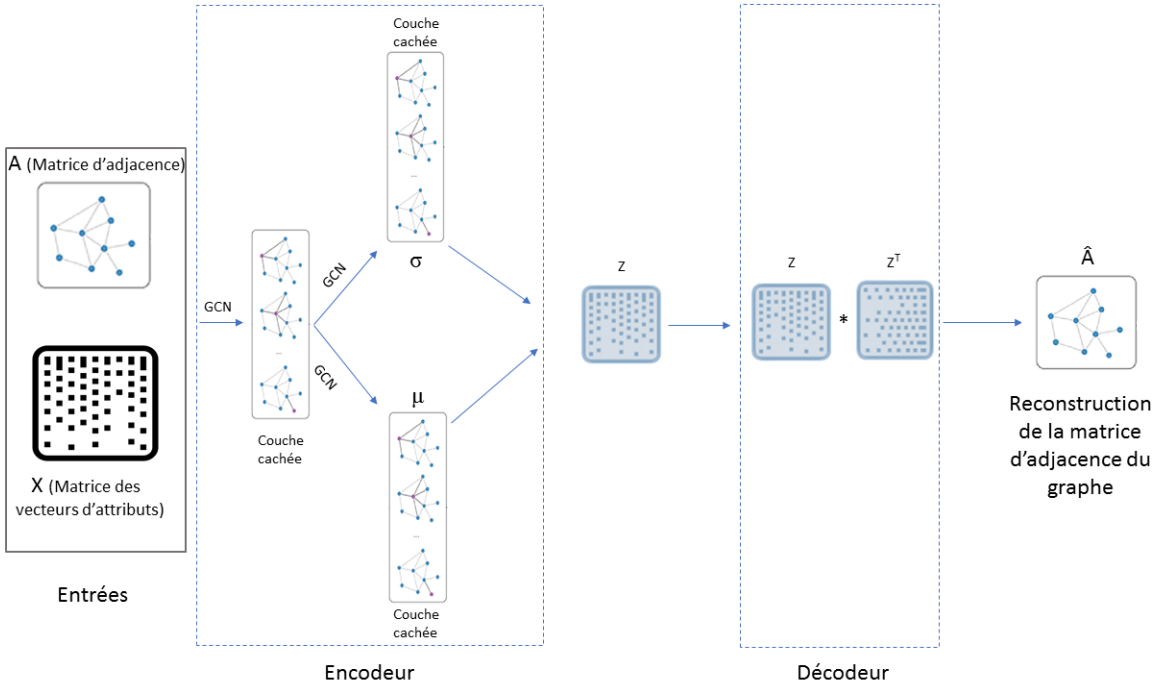


Figure 3.3: Schéma représentatif de l'étape de préentraînement du modèle de clustering profond.

3.2.2 Étape de clustering

Lors de cette étape, il s'agit d'expliciter le processus génératif, qui peut être entraîné dans le cas de l'approche proposée, en maximisant la borne inférieure variationnelle de la vraisemblance logarithmique du graphe (ELBO : Evidence Lower Bound). En considérant l'hyperparamètre K comme nombre de clusters, un cluster c est pris dans la distribution catégorielle $\mathrm{Cat}(\pi)$, celle-ci est paramétrée par π, telle que $\pi \in \mathbb{R}_+^K, et \sum_{k=1}^{K} \pi_k = 1$ désigne la probabilité a priori du cluster. Le processus génératif commence par sélectionner un cluster c_k basé sur la distribution catégorielle [11] $p(c_k)$, celle-ci est paramétrée par $\pi \in \mathbb{R}_+$.

11. Une distribution catégorielle désigne les valeurs qu'une variable aléatoire peut prendre étant donné K clusters possibles, en spécifiant la probabilité de chaque cluster séparément.

$p(c_k)$ est formulée comme suit :

$$p(c_k) = \text{Cat}(c_k \mid \boldsymbol{\pi}) \tag{3.10}$$

L'ensemble des variables latentes $\boldsymbol{Z}$ de la distribution $\mathcal{N}(\mu_c, \sigma_c^2 I)$ est ensuite obtenu en calculant $p(\mathbf{Z} \mid C)$, tel que :

$$p(\mathbf{z}_i \mid c_k) = \mathcal{N}\left(\mathbf{z}_i \mid \boldsymbol{\mu}_{c_k}, \boldsymbol{\sigma}_{c_k}^2 \mathbf{I}\right) \tag{3.11}$$

où z_i indique le nœud v_i de l'espace latent, c_k désigne le cluster k dans l'ensemble des clusters K, μ_{ck} représente la moyenne de la distribution gaussienne, σ_c^2 signifie la variance de la distribution gaussienne, et $\mathbf{I}$ est une matrice identité.

Le décodeur fournit ensuite la sigmoïde du produit interne entre chaque couple de variables latentes. De cette manière, $p(\hat{\boldsymbol{A}} \mid \boldsymbol{Z})$ est paramétré par la partie du décodeur, et suit la loi de Bernoulli selon la définition suivante :

$$\begin{aligned} p(\hat{\boldsymbol{A}} \mid \boldsymbol{Z}) &= \prod_{i=1}^{n} \prod_{j=1}^{n} p\left(\hat{\boldsymbol{A}}_{ij} \mid z_i, z_j\right) \\ p\left(\hat{\boldsymbol{A}}_{ij} \mid z_i, z_j\right) &= \text{Ber}\left(\phi\left(z_i^\top, z_j\right)\right) \end{aligned} \tag{3.12}$$

où $\text{Ber}(\cdot)$ signifie la distribution multidimensionnelle de Bernoulli, $\boldsymbol{Z}$ représente le vecteur latent, ϕ est la fonction d'activation (sigmoïde), et $\hat{\boldsymbol{A}}$ désigne le graphe reconstruit par le décodeur.

De cette manière, la probabilité conjointe $p(\hat{\boldsymbol{A}}, \boldsymbol{Z}, C)$ est factorisé comme suit :

$$p(\hat{\boldsymbol{A}}, \boldsymbol{Z}, C) = p(\hat{\boldsymbol{A}} \mid \boldsymbol{Z}) p(\boldsymbol{Z} \mid C) p(C) \tag{3.13}$$

La fonction de coût du modèle proposé est entraînée afin de maximiser $\mathcal{L}_{elbo}$, celle-ci est définie comme suit :

$$\mathcal{L}_{elbo} = -\mathbb{E}_{q(\boldsymbol{Z},C|X,\boldsymbol{A})}\left[\log\left(\frac{p(\hat{\boldsymbol{A}},\boldsymbol{Z},C)}{q(\boldsymbol{Z},C \mid X,\boldsymbol{A})}\right)\right] \tag{3.14}$$

En remplaçant les expressions de $q(\boldsymbol{Z},C \mid X,\boldsymbol{A})$ et $p(\hat{\boldsymbol{A}},\boldsymbol{Z},C)$ par les définitions du modèle d'inférence (équation **??**), et du modèle génératif (équation **??**) de l'autoencodeur de graphe, la fonction $\mathcal{L}_{elbo}$ est formulé comme suit :

$$\begin{aligned}\mathcal{L}_{elbo} = -\mathbb{E}_{q(\boldsymbol{Z},C|X,\boldsymbol{A})}[&\log(p(\hat{\boldsymbol{A}} \mid \boldsymbol{Z}) + \log(p(\boldsymbol{Z} \mid C)) + \log(p(C)) \\ &- \log(q(\boldsymbol{Z} \mid X,\boldsymbol{A})) - \log(q(C \mid X,\boldsymbol{A}))]\end{aligned} \tag{3.15}$$

En substituant chaque distribution $p(\hat{\boldsymbol{A}} \mid \boldsymbol{Z})$ (équation 3.12), $p(\boldsymbol{Z} \mid C)$ (équation 3.11), $p(C)$ (équation 3.10), $q(\boldsymbol{Z} \mid X,\ \boldsymbol{A})$ (équation 3.3) et $q(C \mid X,\boldsymbol{A})$ (sortie du décodeur) par sa définition, la fonction de coût ELBO est définie comme suit :

$$\mathcal{L}_{elbo} = \mathcal{L}_{rec} + \mathcal{L}_{kld} \tag{3.16}$$

où $\mathcal{L}_{\text{elbo}}$ désigne la fonction de coût ELBO, $\mathcal{L}_{rec}$ représente la fonction de coût de reconstruction, et $\mathcal{L}_{kld}$ signifie la fonction de coût de KL (KL-divergence).

La fonction de coût de reconstruction $\mathcal{L}_{\text{rec}}$ est égale à l'entropie croisée binaire entre le graphe en entrée et le graphe reconstruit :

$$\mathcal{L}_{rec} = \frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{N} a_i \log \mu_a^l|_i + (1-a_i)\log\left(1-\mu_a^l\big|_i\right) \tag{3.17}$$

où n désigne le nombre de nœuds du graphe original $\mathcal{G}$, L représente le nombre d'échantillons de Monte-Carlo [12] dans l'estimateur SGVB [13], $\mu_a^l|_i$ signifie la sortie i du décodeur, et a_i est l'entrée i de la matrice d'adjacence $\boldsymbol{A}$ du graphe original $\mathcal{G}$.

La fonction de coût de KL $\mathcal{L}_{kld}$ (Hui *et al.*, 2020) est responsable du clustering, sa formulation mathématique est la suivante :

$$\begin{aligned}\mathcal{L}_{kld} = &-\frac{1}{2}\sum_{c=1}^{K}\gamma_c\sum_{i=1}^{n}\left(\log\sigma_c^2\big|_i + \frac{\tilde{\sigma}^2|_i}{\sigma_c|_i} + \frac{(\tilde{\mu}|_i - \mu_c|_i)^2}{\sigma_c^2|_i}\right) \\ &+\sum_{c=1}^{K}\gamma_c\log\frac{\pi_c}{\gamma_c} + \frac{1}{2}\sum_{i=1}^{n}\left(1+\log\tilde{\sigma}^2\big|_i\right)\end{aligned} \tag{3.18}$$

où K est le nombre de clusters, n est nombre de nœuds du graphe original $\mathcal{G}$, π_c représente la probabilité a priori du cluster c, γ_c désigne la distribution $q(C \mid X, \hat{\boldsymbol{A}})$, $\tilde{\sigma}^2|_i$ représente le vecteur de variances du cluster c, $\tilde{\sigma}^2|_i$ est la sortie de la deuxième couche de l'encodeur du nœud v_i, $\tilde{\mu}|_i$ est la sortie de la deuxième couche de l'encodeur du nœud v_i, et $\mu_c|_i$ est le vecteur de moyennes du cluster c pour le nœud v_i.

Enfin, la figure 3.4 de la page suivante schématise notre modèle de clustering décrit dans cette partie.

12. Monte-Carlo est une technique d'échantillonnage aléatoire d'une distribution de probabilité. Dans notre cas, elle vise à rassembler des échantillons pour approximer la distribution la fonction de coût ELBO.

13. L'estimateur du gradient stochastique variationnel de Bayes (Stochastic Gradient Variational Bayes : SGVB) fait référence à l'approximation d'intégrales en utilisant l'inférence bayésienne. L'estimateur vise à approximer une fonction de coût avec de nombreux échantillons aléatoires (généré par Monte-Carlo dans notre cas).

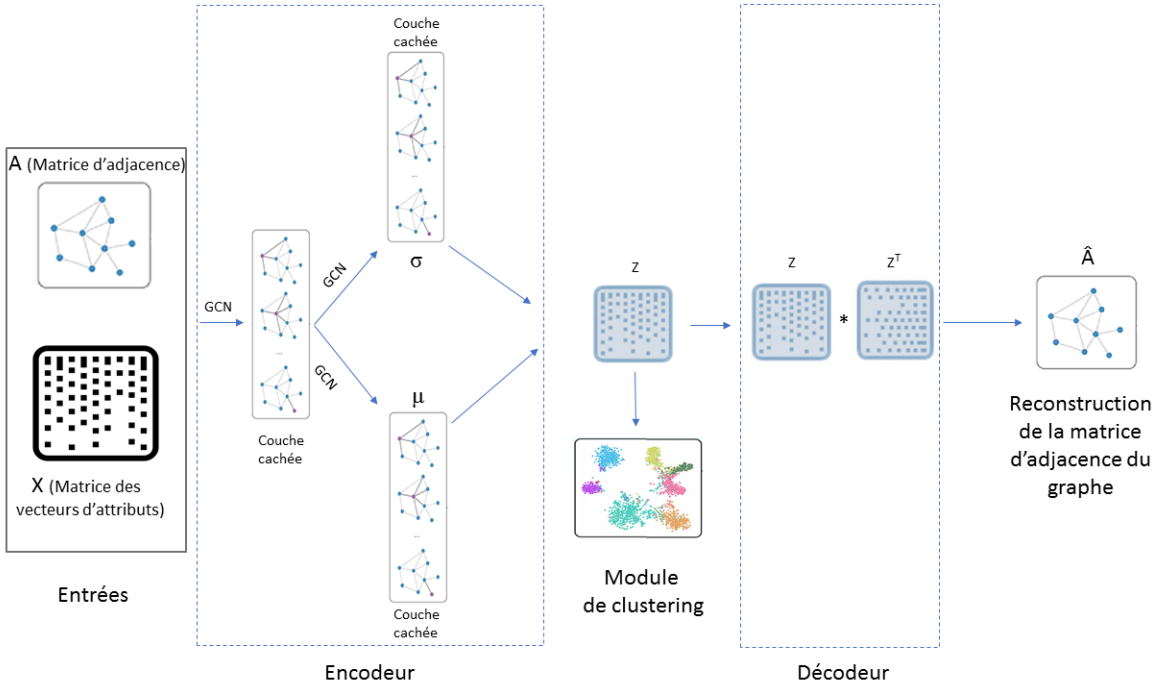


Figure 3.4: Schéma représentatif de l'étape de clustering du modèle de clustering profond proposée.

3.3 Phase de clustering profond dynamique du graphe attribué

Dans cette phase, nous présentons notre approche de clustering profond dynamique D-DGC, ce dernier est composé du modèle de clustering profond des graphes attribués (voir la section 3.2), ainsi qu'un module d'apprentissage dynamique. L'idée principale est de se baser sur les nœuds centroïdes formé avec le modèle de clustering profond des graphes attribués, afin de réduire le caractère arbitraire des descripteurs et améliorer la formation des clusters finaux.

D-DGC prend en entrée la matrice d'adjacence $\boldsymbol{A}$, et la matrice des attributs $\boldsymbol{X}$ des nœuds du graphe $\mathcal{G}$. L'encodage est ensuite enclenché, il s'agit de compresser les données en entrées en un ensemble de variables latentes $\boldsymbol{Z}$ en utilisant des couches de convolutions du GCN. L'ensemble des variables $\boldsymbol{Z}$ est ensuite utilisé

pour trouver les variables latentes z_i, des nœuds v_i associés aux nœuds centroïdes μ_c. À la base, les nœuds centroïdes du graphe $\mathcal{G}$ sont reconstruits par le décodeur, ce qui ne représente pas forcément des nœuds réellement présents dans le graphe original. Pour remédier à cela, le premier voisin le plus proche d'un nœud centroïde μ_c dans l'espace latent $\boldsymbol{Z}$ remplace ce dernier. Dans ce contexte, nous définissons une affectation souple de clustering q_{ic}, celle-ci prend en compte la probabilité d'affecter un nœud de l'espace latent z_i au nœud centroïde μ_c. Ainsi q_{ic} est définie comme suit :

$$q_{ic} = \frac{\left(1 + \frac{\|z_i - \mu_c\|^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}}{\sum_{c'} \left(1 + \frac{\|z_i - \mu_{c'}\|^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}} \tag{3.19}$$

où μ_c désigne le nœud centroïde du cluster c, et z_i représente la variable latente du nœud v_i.

Pour chaque nœud en entrée v_i, nous définissons $h_{ic'}$ comme la valeur maximale que peut prendre q_{ic}. Par exemple, h_{i1} est la valeur maximale de q_{ic} et h_{i2} est la deuxième valeur maximale de q_{ic}. Mathématiquement, $h_{ic'}$ peut être formulé comme suit :

$$h_{ic'} = \begin{cases} \max_c \{q_{ic}\}, \text{ si } c' = 1 \\ \max_c \left\{q_{ic} \mid q_{ic} < h_{i(c'-1)}\right\}, \text{ sinon} \end{cases} \tag{3.20}$$

où q_{ic} désigne l'affectation souple du nœud v_i au cluster c.

Ainsi, nous prenons en compte deux principaux groupes de nœuds, les nœuds conflictuels et les nœuds non conflictuels. Les nœuds conflictuels ont de faibles affectations souples de clustering q_{ik}, ils sont situés dans les limites de chaque

cluster, et ont des affectations souples très proches les unes des autres ; ce qui complique l'identification de la ressemblance de chaque nœud conflictuel à son centre de cluster correspondant. Par ailleurs, un nœud non conflictuel est caractérisé par une condition qui contrôle le niveau d'incertitude d'appartenance à un cluster (seuil de la probabilité d'appartenance). Dans ce contexte, un nœud conflictuel est défini comme ayant : (1) un seuil inférieur de probabilité d'appartenance λ_1, et (2) la différence entre ses deux probabilités d'appartenance les plus élevées est inférieure au seuil λ_2. Ce dernier appartient à l'ensemble $\bar{E}$, qui est défini par l'équation suivante :

$$\bar{E} = \{v_i \in V \mid h_{i1} < \lambda_1 \text{ ou } (h_{i1} - h_{i2}) < \lambda_2\} \tag{3.21}$$

où v_i représente le nœud en entrée, V représente l'ensemble des nœuds du graphe $\mathcal{G}$, h_{i1} est la valeur maximale de q_{ic}, h_{i2} est la deuxième valeur maximale de q_{ic}, λ_1 représente le seuil de confiance minimal en dessous duquel un nœud est considéré comme étant en conflit, et λ_2 désigne la différence minimale entre la probabilité d'affectation la plus élevée et la deuxième probabilité la plus élevée.

La formulation de λ_1 et λ_2, les deux hyperparamètres de contrôle, est décrite ci-dessous :

$$\lambda_1 = \frac{\alpha}{K} \text{ et } \lambda_2 = \frac{\lambda_1}{2}, \quad \alpha \in [1, K] \tag{3.22}$$

où α désigne le seuil de confiance défini à la base (par expérimentation), et K désigne le nombre de clusters.

Ainsi, D-DGC est initialisé de manière à garantir que λ_1 et λ_2 reçoivent des valeurs élevées, afin de favoriser le processus de reconstruction initialement, et

permettre d'avoir le maximum de nœuds non conflictuels. Ce qui induit que la majeure partie de l'ensemble d'apprentissage est consacrée à la reconstruction et le reste est utilisé pour effectuer la construction des centroïdes. Progressivement, l'importance de la fonction de coût de reconstruction issue de l'autoencodeur de graphe diminue, afin d'accroitre l'importance du processus de reconstruction basé sur les centroïdes de clusters. En effet, l'élimination brutale de la reconstruction classique de l'autoencodeur de graphe accroit le phénomène du caractère arbitraire des descripteurs, ce qui est opposé à l'objectif principal qui est de le réduire. La mesure de confiance appliquée pour calculer la fonction de coût du modèle dynamique est divisée en deux. Les nœuds non conflictuels sont sélectionnés pour la construction du centroïde, et ceux en conflit sont utilisés par la fonction de coût de reconstruction de l'autoencodeur, jusqu'à ce que le modèle identifie de manière efficace leurs centres correspondants.

α est mis à jour graduellement afin de capturer la dynamique des nœuds conflictuels. La dynamique de la fonction de coût est décrite par τ, qui spécifie la quantité de reconstruction et indique la progression de l'entraînement. τ est défini comme suit :

$$\tau = \frac{|\bar{E}|}{n} \tag{3.23}$$

où $\bar{E}$ représente l'ensemble des nœuds conflictuels, et n désigne le nombre de nœuds du graphe original $\mathcal{G}$.

Au fur et à mesure que l'entraînement du modèle progresse, la fonction de coût dynamique atteindra la stabilité et le nombre de nœuds conflictuels diminuera. Ceci est dû premièrement à la mise à jour graduelle (diminution) des valeurs des hyperparamètres λ_1 et λ_2, et en second lieu la mise à jour des centroïdes, qui

stabilisera la fonction de coût dynamique. De ce fait, notre modèle (D-DGC) est entraîné jusqu'à ce que λ_1 et λ_2 auront des valeurs négligeables (proches de zéro) et qu'il ne reste plus aucune reconstruction.

Ainsi, la fonction de coût du D-DGC est définie comme suit :

$$\mathcal{L}_{D-DGC} = \begin{cases} \mathcal{L}_{rec}, \text{ si } v_i \in \bar{E} \\ \mathcal{L}_{rec} + \mathcal{L}_{kld}, \text{ sinon} \end{cases} \tag{3.24}$$

Où, à titre de rappel, v_i désigne le nœud i, $\bar{E}$ représente l'ensemble des nœuds conflictuels, $\mathcal{L}_{rec}$ désigne la fonction de coût de reconstruction (défini dans l'équation 3.17), et $\mathcal{L}_{kld}$ signifie la fonction de coût du clustering (défini dans l'équation 3.18). À la fin du processus d'apprentissage, chaque nœud sera attribué au cluster dont le centroïde a été calculé grâce au processus dynamique décrit lors de cette section.

À titre indicatif, nous présentons la figure 3.5 de la page suivante, qui vise à schématiser la solution D-DGC.

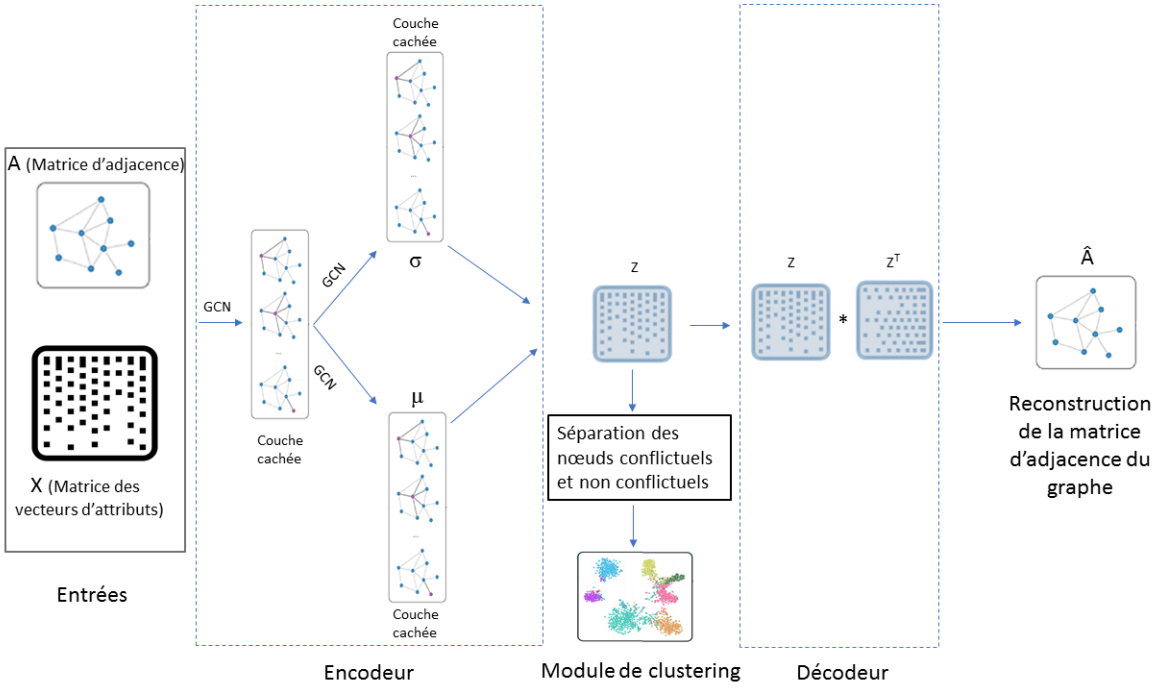


Figure 3.5: Schéma représentatif du modèle proposé D-DGC.

3.4 Conclusion

Dans ce chapitre, nous avons présenté notre approche D-DGC. Cette dernière est basée principalement sur une phase de clustering profond des graphes attribués, qui permet la distinction entre les clusters de nœuds au sein d'un graphe attribué. La phase de clustering dynamique vise, quant à elle, à utiliser une fonction de coût dynamique, afin de diminuer l'effet du caractère arbitraire des descripteurs présent dans la plupart des solutions déjà proposées basées sur les autoencodeurs de graphes. Dans le prochain chapitre, nous présenterons les résultats obtenus lors des expérimentations sur les données de références, citées dans la plupart des travaux de la littérature dans le clustering profond des graphes attribués.

CHAPITRE IV

ÉVALUATION DE L'APPROCHE PROPOSÉE

Ce chapitre présente une évaluation empirique de la solution D-DGC sur des ensembles de données de références réelles. La performance de l'approche proposée est comparée à différentes solutions de clustering traditionnelles, et de clustering profond des graphes attribués basés sur les autoencodeurs de graphes. Nous entamerons ce chapitre par la description du cadre expérimental ainsi que les critères de performances retenus, ensuite nous aborderons l'énumération des résultats obtenus et l'analyse de ces derniers.

4.1 Cadre expérimental

4.1.1 Algorithmes sélectionnés pour la comparaison

Afin d'illustrer les performances de l'approche proposée, nous avons sélectionné plusieurs méthodes décrites dans l'état de l'art, à savoir les méthodes traditionnelles et les méthodes de clustering profond des graphes attribués. Nous comparons notre solution aux algorithmes K-means et au clustering spectral (Spectral Clustering) (Ng *et al.*, 2002) premièrement, ceux-ci utilisent uniquement le contenu des nœuds du graphe (les vecteurs d'attributs des nœuds seulement). Par la suite, il sera question de comparer notre approche D-DGC avec des modèles traitant uniquement la structure (la matrice d'adjacence du graphe) notamment avec le

clustering spectral, DeepWalk (Perozzi *et al.*, 2014), DNGR (Cao *et al.*, 2016), et M-NMF (Wang *et al.*, 2017b). Enfin, il sera question de comparer D-DGC aux solutions utilisant le contenu des nœuds du graphe (les vecteurs d'attributs associés aux nœuds) et la structure du graphe (la matrice d'adjacence du graphe), tel que GAE (Kipf et Welling, 2016), VGAE (Kipf et Welling, 2016), ARGA(Pan *et al.*, 2018), ARVGA(Pan *et al.*, 2018), DAEGC(Wang *et al.*, 2019), et GMM-VGAE (Hui *et al.*, 2020).

4.1.2 Ensembles de données de référence utilisés pour la comparaison

Nous comparons les algorithmes décrits dans la section précédente sur 6 ensembles de données citées et utilisées dans la littérature. Une description de ces données est fournie dans ce qui suit :

1. Cora (Rossi et Ahmed, 2015) est un ensemble de données qui se compose de 2 708 publications scientifiques (nœuds), liées par 5 429 liens et regroupées dans 7 clusters. Chaque publication (nœud) englobe un vecteur d'attributs (mot ayant comme valeur 0 ou 1) indiquant l'absence, ou la présence du mot correspondant dans le dictionnaire de 14 333 mots uniques.
2. Citeseer (Rossi et Ahmed, 2015) est un ensemble de données formé de 3 312 publications scientifiques (nœuds) regroupées dans 6 clusters. Le graphe de citations dispose de 4 732 liens, chaque publication (nœud) englobe un vecteur d'attributs (mot ayant comme valeur 0 ou 1) indiquant l'absence, ou la présence du mot correspondant dans le dictionnaire de 3 703 mots uniques.
3. Pubmed (Rossi et Ahmed, 2015) contient 19 717 publications scientifiques (nœuds) de la base de données PubMed relatives au diabète regroupé dans 3 clusters différents. Le graphe de citations se compose de 44 338 liens.

Chaque publication (nœud) englobe un vecteur de mot pondéré TF/IDF (vecteur d'attributs) provenant d'un dictionnaire composé de 500 mots uniques.

4. L'ensemble de données du trafic aérien brésilien (Brazil-air-traffic) regroupe des données recueillies auprès de l'agence nationale de l'aviation civile (ANAC) [14] de janvier à décembre 2016. Le graphe compte 131 nœuds, 1 038 liens, et 4 clusters.
5. L'ensemble de données du trafic aérien européen (Europe-Air-traffic) réunies les données collectées auprès de l'office statistique de l'Union européenne (Eurostat) [15] de janvier à novembre 2016. Le graphe contient 399 nœuds, liés par 5 995 liens, et regroupés en 4 clusters. L'activité aéroportuaire est mesurée par le nombre total d'atterrissages et de décollages au cours de l'année correspondante.
6. L'ensemble de données du réseau de trafic aérien américain (USA-Air-traffic) correspond à des données collectées par le Bureau of Transportation Statistics [16] de janvier à octobre 2016. Le réseau compte 1 190 nœuds, liés par 13 599 liens, et regroupés en 4 clusters. L'activité aéroportuaire est mesurée par le nombre total de personnes qui sont passées (au départ et à l'arrivée) par l'aéroport au cours de la période correspondante.

4.1.3 Critères d'évaluation du clustering profond des graphes attribués

Afin d'évaluer la performance des approches sélectionnées pour la comparaison, nous utilisons trois métriques standards citées et appliquées dans la littérature.

14. https ://www.anac.gov.br/
15. https ://ec.europa.eu/
16. https ://transtats.bts.gov/

La première métrique est l'exactitude du clustering (ACC : Accuracy) (Huang et Ling, 2005), celle-ci calcule le taux de clustering correct moyen, où l'ensemble des étiquettes identifiées pour un nœud doit correspondre exactement à l'ensemble d'étiquettes de vérité de terrain (*ground truth*). L'exactitude admet des valeurs entre 0 et 1, ce qui veut dire qu'une valeur proche de 0 désigne une mauvaise correspondance entre les clusters identifiés et la répartition originale, tandis qu'une valeur proche de 1 désigne une forte correspondance entre elles. L'exactitude (ACC) est définie comme suit :

$$ACC(Y, C) = \max_{m \in P} \frac{1}{n} \sum_{i=1}^{n} 1(m(\hat{y}_i) = y_i) \tag{4.1}$$

où y_i est l'étiquette du nœud v_i de l'ensemble de données de référence, c_k est l'affectation du cluster C générée par le modèle de clustering, P désigne l'ensemble de toutes les permutations dans $[1; K]$ (K désigne le nombre de clusters dans ce cas), et m est une fonction de correspondance qui s'étend sur toutes les correspondances possibles entre les affectations de clusters C et des étiquettes de vérité de terrain Y.

La deuxième métrique est le taux d'information mutuelle normalisée (NMI : Normalized Mutual Information) (Mogotsi, 2010), celle-ci calcule les informations mutuelles normalisées entre les clusters identifiés et les étiquettes de vérité de terrain. Le taux d'information mutuelle normalisée admet des valeurs entre 0 (aucune information mutuelle) et 1 (corrélation parfaite), et est défini comme suit :

$$NMI(Y, C) = \frac{2I(Y, C)}{[H(Y) + H(C)]} \tag{4.2}$$

où Y désigne les étiquettes de vérité de terrain, C désigne les étiquettes des clusters

identifiés, $I(Y, C) = H(Y) - H(Y \mid C)$ est l'information mutuelle entre Y et C, et H désigne l'entropie conditionnelle de Y sachant C.

La troisième métrique est l'indice de Rand ajusté (ARI : Adjusted Rand Index) (Hubert et Arabie, 1985), celle-ci calcule une mesure de similarité entre les clusters identifiés et les clusters réels, en considérant toutes les paires de nœuds, et en comptant les paires qui sont attribuées dans les clusters correspondants ou différents. L'indice de Rand ajusté admet une valeur proche de 0 lors d'une affectation de clustering aléatoire indépendamment du nombre de clusters et de nœuds, ou exactement 1 lorsque les clusters sont identiques (jusqu'à une permutation). Ainsi le ARI est défini comme suit :

$$ARI(Y, C) = \frac{2(ad - bc)}{(a + b)(b + d) + (a + c)(c + d)} \tag{4.3}$$

où a, b, c, et d dénotent respectivement, le nombre de paires affectées au même cluster dans Y et C ; dans le même cluster dans Y, mais pas dans C ; dans le même cluster dans C, mais pas dans Y ; et dans différents clusters dans Y et C.

4.1.4 Critère d'évaluation du caractère arbitraire des descripteurs

Afin d'évaluer le caractère arbitraire des descripteurs, nous introduisons la métrique Δ_{CA} (Mrabah *et al.*, 2020), qui est caractérisé par la déviation du gradient de la fonction de coût supervisé réelle par rapport aux paramètres du réseau de neurones (l'autoencodeur de graphe dans notre cas) w après l'introduction des pseudoétiquettes. Mathématiquement, le caractère arbitraire des descripteurs peut être exprimé comme le cosinus de l'angle entre le gradient de la fonction objectif supervisée réelle et le gradient de la fonction objectif non supervisée. La caractérisation Δ_{CA} admet une valeur entre 0 (caractère arbitraire élevé) et 1

(caractère arbitraire nulle).

Δ_{CA} est décrit par l'équation suivante :

$$\Delta_{CA} = \cos\left(\frac{\partial \mathcal{L}\left(v, y_{ref}, w\right)}{\partial w}, \frac{\partial \mathcal{L}\left(v, y_{pseudo}, w\right)}{\partial w}\right) \tag{4.4}$$

où v représente le nœud en entrée, $\mathcal{L}$ est la fonction de coût du modèle , y_{ref} est le vecteur des étiquettes de l'ensemble de données de référence (vérité de terrain), et y_{pseudo} est le vecteur des pseudoétiquettes.

4.2 Expérimentations sur les données de références

4.2.1 Déroulement des expérimentations

Nous commençons par reconduire les expérimentations sur les modèles de clustering K-means et le clustering spectral concernant les approches traditionnelles. Ensuite, pour les approches de clustering profond des graphes attribués, nous utilisons les implémentations mises à dispositions par les auteurs des modèles : (1) DeepWalk, (2) DNGR, (3) M-NMF, (4) GAE, (5) VGAE, (6) ARGA, (7) ARVGA. Concernant les approches DAEGC et GMM-VGAE, nous avons implémenté les deux solutions selon les indications de l'article original de chaque approche (implémentation publique non disponible). Il est à noter également que pour toutes les approches citées dans cette partie, nous avons reconduit les mêmes paramètres précisés dans les articles originaux, de telle manière à avoir des résultats corrects pour chaque ensemble de données utilisé.

4.2.2 Paramétrage de la solution D-DGC

Concernant la solution D-DGC, nous avons établi dans le tableau 4.1 un ensemble de paramètres avec leurs valeurs précises.

Tableau 4.1: Configuration des paramètres de D-DGC.

Paramètre	**Valeur**
Dimension de la première couche GCN	32
Dimension de la deuxième couche GCN	16
Nombre d'itérations dans le préentraînement	200
Optimiseur du préentraînement	Adam
Taux d'apprentissage du préentraînement	0.01
Nombre d'itérations de l'entraînement	200
Optimiseur de l'entraînement	Adam
Taux d'apprentissage de l'entraînement	0.01

Les ensembles de données utilisés admettent une configuration des hyperparamétres de contrôle de la fonction de coût dynamique. De ce fait, nous présentons dans le tableau 4.2 ci-dessous la configuration utilisée pour chaque ensemble de données de référence, comportant les hyperparamétres λ_1, λ_2, et la fréquence de mise à jour de ces derniers représenté par M.

Tableau 4.2: Configuration des hyperparamétres de D-DGC sur chaque ensemble de données de référence.

Méthode	λ_1	λ_2	M
Cora	0.3	0.15	10 itérations
Citeseer	0.2	0.1	1 itération
Pubmed	0.4	0.2	5 itérations
USA-Air-Traffic	0.3	0.15	1 itération
Europe-Air-Traffic	0.01	0.005	1 itération
Brazil-Air-Traffic	0.25	0.12	1 itération

4.2.3 Reproductivité des expérimentations

Afin de reproduire les expérimentations sur les modèles cités dans la section 4.1.1, nous détaillons dans cette partie la configuration matérielle et logicielle de ce travail. Concernant la configuration matérielle, nous avons utilisé un serveur contenant un CPU Intel(R) Xeon(R) E5-2620 V4 @ 2.10GHz. Concernant la partie logicielle, le serveur dispose d'un système d'exploitation Ubuntu (version : 18.04.5 LTS) sur lequel les modèles ont été implémentés avec le langage Python (version : 3.7.4), ainsi que les librairies PyTorch (version : 1.3.1), et Sklearn (version : 0.23.1).

4.3 Résultats et discussion

L'objectif de cette section est d'analyser les résultats de D-DGC sur différents ensembles de données de références : (1) analyse de ACC, NMI et ARI de D-DGC par rapport aux approches de clustering des graphes attribués (méthodes traditionnelles et méthodes basées sur l'apprentissage profond) ; (2) comparaison des espaces latents de l'approche proposée avec l'approche GMM-VGAE, (3) analyse du nombre de nœuds conflictuels et non conflictuels pour l'approche D-DGC, et (4) analyse de l'impact du caractère arbitraire des descripteurs entre GMM-VGAE et D-DGC.

Le premier aspect étudié dans nos tests est la performance de clustering de D-DGC par rapport aux modèles de l'état de l'art. Les tableaux 4.3 et 4.4 reportent les entrées utilisées pour chaque approche ; celles-ci sont catégorisées en trois groupes : (1) 'C' fait référence au contenu c'est-a-dire les attributs des nœuds du graphe, (2) 'S' fait référence à la structure c'est-à-dire la matrice d'adjacence du graphe, et (3) 'C&S' fait référence à l'utilisation de la structure et du contenu. Il décrit aussi les performances de chaque modèle, inscrites en pourcentage (%), et évaluées par

les métriques ACC, NMI, et ARI. Il est à rappeler que pour chaque algorithme et chaque ensemble de données de référence, la valeur inscrite sur le tableau est la valeur maximale des résultats obtenus sur 10 itérations (même démarche utilisée dans l'état de l'art).

Tableau 4.3: Tableau comparatif des performances de clustering entre différentes méthodes de clustering de graphes attribués sur les ensembles de données de référence Cora, Citeseer, et Pubmed.

Méthode	**Entrée**	**Cora**			**Citeseer**			**Pubmed**		
		ACC	**NMI**	**ARI**	**ACC**	**NMI**	**ARI**	**ACC**	**NMI**	**ARI**
k-means	C	50.3	31.7	24.4	54.4	31.2	28.5	58.0	27.8	24.6
Spectral	C	34.7	14.7	07.1	44.1	20.3	18.3	60.2	30.9	27.7
Spectral	S	34.2	19.5	04.5	25.9	11.8	01.3	52.8	09.7	06.2
DeepWalk	S	48.4	32.7	24.3	33.7	08.9	09.2	54.3	10.2	08.8
DNGR	S	41.9	31.8	14.2	32.6	18.0	04.3	46.8	15.3	05.9
M-NMF	S	42.3	25.6	16.1	33.6	09.9	07.0	47.0	08.4	05.8
GAE	C&S	61.3	44.4	38.1	48.2	22.7	19.2	63.2	24.9	24.6
VGAE	C&S	64.7	43.4	37.5	51.9	24.9	23.8	69.6	28.6	31.7
ARGA	C&S	64.0	44.9	35.2	57.3	35.0	34.1	68.1	27.6	29.1
ARVGA	C&S	63.8	45.0	37.4	54.4	26.1	24.5	63.5	23.2	22.5
DAEGC	C&S	70.4	52.8	49.6	67.2	39.7	41.0	67.1	26.6	27.8
GMM-VGAE	C&S	71.5	53.1	47.4	67.5	40.7	42.4	71.1	29.9	33.0
D-DGC	C&S	**75.4**	**56.0**	**55.3**	**68.8**	**42.2**	**44.3**	**73.1**	**32.3**	**36.2**

Dans le tableau 4.3, les résultats révèlent que les méthodes traitant uniquement du contenu des nœuds du graphe (vecteurs d'attributs des nœuds) ou uniquement de la structure du graphe (matrice d'adjacence du graphe) peinent à bien identifier les clusters. Ceci est dû aux manques de données complémentaires que peut fournir la structure du graphe et le contenu des attributs des neuds en même temps. Cependant, les algorithmes de clustering profond des graphes attribués utilisant des autoencodeurs de graphes permettent de mieux identifier les clusters, ceci est dû à l'incorporation des données du graphe ainsi que les attributs des nœuds, ce qui procure un avantage considérable par rapport aux méthodes traditionnelles

variant entre 15% et 20% de différence dans l'ensemble de données Cora, 15% dans Citeseer, et entre 5% et 15% dans l'ensemble de données Pubmed. Concernant notre modèle (D-DGC), ce dernier offre les meilleurs résultats pour toutes les métriques et pour tous les ensembles de données de référence. Ceci est dû aux faibles taux de nœuds conflictuels identifié pour chaque cluster.

Tableau 4.4: Tableau comparatif des performances de clustering entre différentes méthodes de clustering de graphes attribués sur les ensembles de données de référence USA Air-Traffic, Europe Air-Traffic, et Brazil Air-Traffic.

Méthode	**Entrée**	**USA Air-Traffic**			**Europe Air-Traffic**			**Brazil Air-Traffic**		
		ACC	**NMI**	**ARI**	**ACC**	**NMI**	**ARI**	**ACC**	**NMI**	**ARI**
Kmeans	C	42.5	22.6	16.0	32.8	09.8	01.9	41.2	28.5	10.0
Spectral	C	29.5	09.1	01.5	29.3	03.4	00.6	32.8	11.7	01.1
Spectral	S	35.4	08.2	04.9	27.3	03.9	00.1	33.6	06.1	01.0
DeepWalk	S	37.4	10.0	04.7	30.8	03.8	01.3	35.9	08.3	05.1
DNGR	S	45.8	15.6	12.8	43.1	22.8	17.2	42.7	12.3	10.8
M-NMF	S	38.2	25.8	09.6	40.9	26.9	17.7	49.6	38.9	25.9
GAE	C&S	43.9	13.6	11.8	47.6	19.9	12.7	62.6	37.8	30.8
VGAE	C&S	45.8	23.6	15.7	49.9	23.5	16.7	64.1	38.0	30.7
ARGA	C&S	48.5	25.1	17.6	50.1	23.7	16.6	67.2	43.8	38.0
ARVGA	C&S	48.2	24.7	**20.0**	51.1	22.2	16.2	63.4	43.8	38.5
DAEGC	C&S	46.4	**27.2**	18.4	53.6	30.9	23.3	71.0	47.4	41.2
GMM-VGAE	C&S	48.1	21.9	13.2	51.1	27.5	21.7	70.2	46.0	41.9
D-DGC	C&S	**50.2**	24.2	16.9	**56.9**	**32.3**	**26.3**	**72.5**	**47.6**	**44.3**

Le tableau 4.4 étudie quant à lui les ensembles de données de référence USA-Air-Traffic, Brazil-Air-Traffic et Europe-Air-Traffic. Les résultats affichés indiquent que les méthodes de clustering profond des graphes attribués basées sur les autoencodeurs de graphes donnent de meilleurs résultats que ceux utilisant la structure du graphe seulement, ou le contenu des vecteurs d'attributs des nœuds uniquement. Par ailleurs, D-DGC obtient les meilleures performances en ce qui concerne la métrique ACC pour tous les ensembles de données ; les meilleurs NMI et ARI pour les ensembles de données Europe-Air-Traffic et Brazil-Air-Traffic. Ceci in-

dique que même pour des ensembles de données où le nombre de nœuds et de liens est réduit, D-DGC offre des performances supérieures par rapport aux modèles de clustering traditionnels et de clustering profond basé sur l'autoencodeur de graphes ; grâce à une bonne identification des nœuds conflictuels.

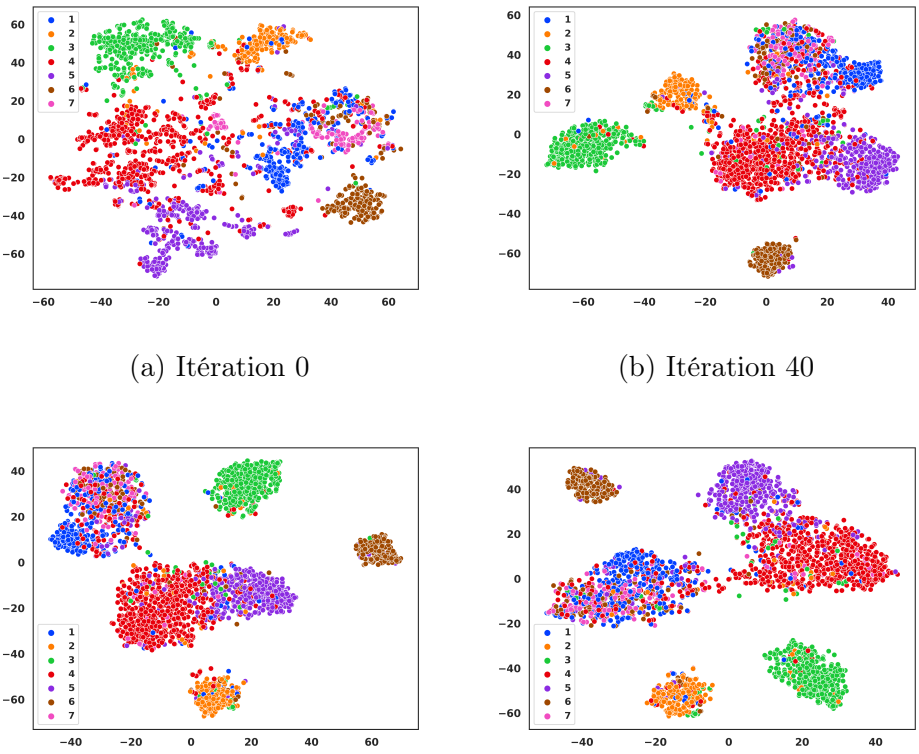
Dans l'ensemble, GMM-VGAE est le modèle ayant les meilleures performances de clustering profond avec une fonction de coût statique. Cependant, notre modèle (D-DGC) obtient la meilleure performance par rapport à tous les modèles de clustering profond en utilisant une fonction de coût dynamique. Pour cela, nous prenons en compte uniquement les modèles GMM-VGAE et D-DGC pour la suite des analyses. Ces derniers seront testés avec l'ensemble de données de référence Cora pour l'analyse des espaces latents ; puis, avec Cora, Citeseer, et pubmed pour le suivi de l'évolution des nœuds conflictuels et non conflictuels ; ainsi que pour l'impact du caractère arbitraire des descripteurs.

Le deuxième aspect étudié est le suivi du développement des représentations latentes pendant l'entraînement de GMM-VGAE (figure 4.1) et D-DGC (figure 4.2), sur l'ensemble de données de référence Cora, en utilisant l'algorithme t-SNE [17].

La figure 4.1 décrit le développement des espaces latents pendant l'entraînement du modèle GMM-VGAE sur l'ensemble de données de référence Cora. Celui-ci est caractérisé par un ensemble de 7 clusters. Nous constatons dans les itérations 0 et 40 une mauvaise identification, car les clusters apparaissent très proches les uns des autres. Concernant les itérations 80 et 120, on constate une légère amélioration due à la minimisation de la fonction de coût (composé de reconstructions de l'autoencodeur et de KL). Néanmoins, les clusters 1, 4 et 7 ne sont pas bien identifiés, ceci est dû à un caractère arbitraire des descripteurs élevé, et à un

17. t-SNE est un algorithme qui permet de réduire la dimensionnalité des données, en représentant l'ensemble de données de grande dimension en un ensemble à deux ou trois dimensions.

grand nombre de nœuds conflictuels mal identifiés.



(a) Itération 0

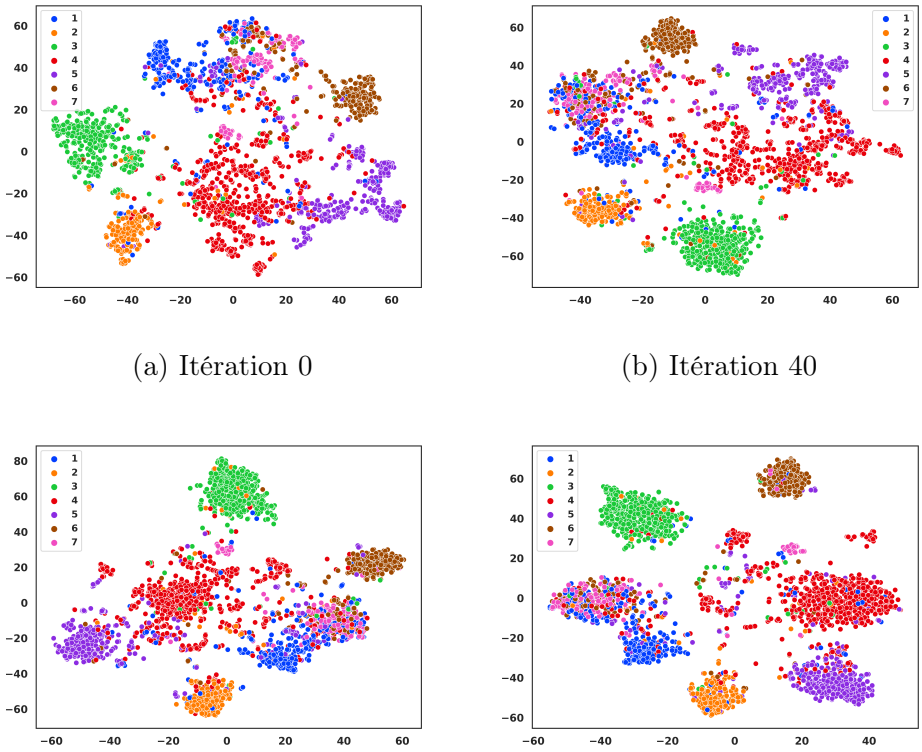
(b) Itération 40

(c) Itération 80

(d) Itération 120

Figure 4.1: Visualisations des représentations latentes de GMM-VGAE sur Cora en utilisant t-SNE.

Cependant, dans la figure 4.2, nous constatons une meilleure identification des 7 clusters de l'ensemble de données de référence Cora avec le modèle D-DGC. Au niveau des itérations 0 et 40, l'identification est similaire à celle de la figure 4.1, mais les itérations 80 et 120 offrent une meilleure identification grâce à la fonction de coût dynamique, qui réduit petit à petit le nombre de nœuds conflictuels en faveur des nœuds non conflictuels. De même, la réduction du caractère arbitraire des descripteurs a permis à D-DGC d'avoir des performances supérieures par rapport à celle du GMM-VGAE.



(a) Itération 0

(b) Itération 40

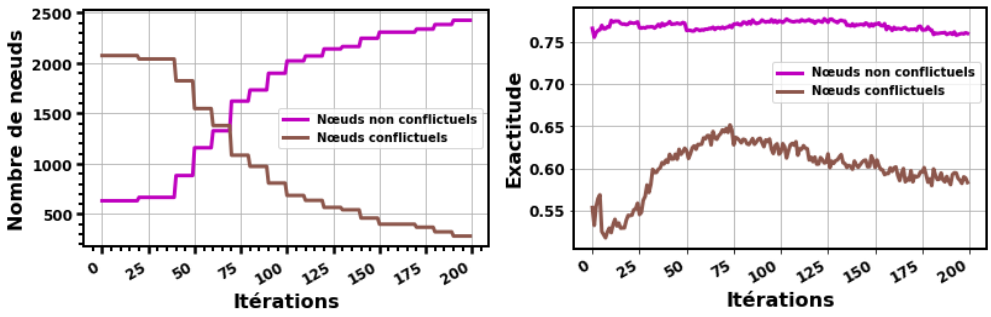
(c) Itération 80

(d) Itération 120

Figure 4.2: Visualisations des représentations latentes de D-DGC sur Cora en utilisant t-SNE.

Pour mieux interpréter les résultats précédents, nous présentons les figures 4.3, 4.4, et 4.5 qui illustrent la dynamique d'apprentissage pendant l'entraînement de D-DGC sur Cora, Citeseer et Pubmed. Celles-ci comparent le nombre de nœuds en conflit au nombre de nœuds non conflictuels (partie (a) des figures), ainsi que l'exactitude des nœuds en conflits par rapport aux nœuds non conflictuels (partie (b) des figures).

Dans la figure 4.3 (a), nous constatons que le nombre de nœuds non conflictuels au départ est de 632, au contraire des nœuds conflictuels qui est de 2076 nœuds. Pendant l'entraînement de D-DGC, le nombre de nœuds non conflictuels augmente



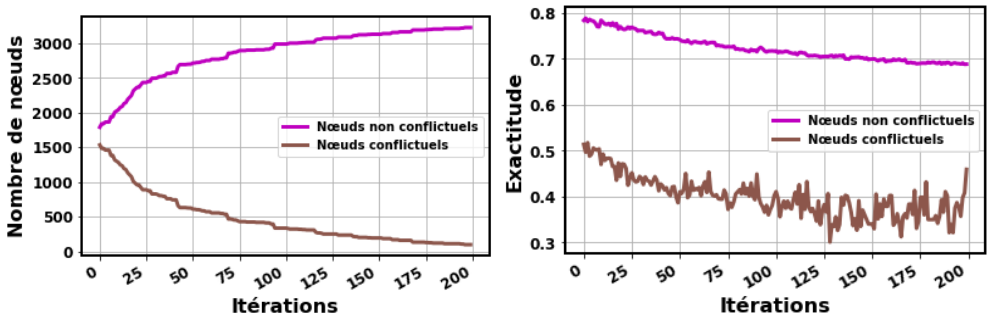
(a) (b)

Figure 4.3: Courbes représentatives de l'évolution des nœuds ainsi que leurs exactitudes en utilisant D-DGC avec l'ensemble de données de référence Cora.

périodiquement et constamment dû à la mise à jour des hyperparamètres λ_1 et λ_2, ce qui provoque la baisse du nombre de nœuds conflictuels. À l'arrivée des 200 itérations (fin de l'entraînement du modèle), près de 2427 nœuds non conflictuels sont bien séparés, alors que 281 nœuds restent conflictuels. Dans la figure 4.3 (b), l'exactitude des nœuds non conflictuels fluctue à plus de 0.75 tout au long de l'apprentissage de D-DGC. Cependant, l'exactitude des nœuds conflictuels démarre à 0.55, puis fluctue entre 0.5 et 0.65 tout au long de l'apprentissage. Ceci explique, la qualité des performances obtenues dans le tableau 4.3 concernant l'ensemble de données de référence Cora pour D-DGC.

Dans la figure 4.4 (a), le nombre de nœuds non conflictuels au départ est de 1791 nœuds, alors que le nombre de nœuds conflictuels est de 1536. Pendant l'entraînement de D-DGC, le nombre de nœuds non conflictuels augmente constamment, ce qui provoque la baisse du nombre de nœuds conflictuels. À la fin de l'entraînement du modèle, 3229 nœuds non conflictuels sont bien séparés, alors que 98 nœuds restent conflictuels. Dans la figure 4.4 (b), l'exactitude des nœuds non conflictuels démarre à 0.8 et diminue progressivement tout au long de l'apprentissage de D-DGC jusqu'à l'atteinte d'un peu moins de 0.7. Cependant, l'exactitude des nœuds conflictuels démarre à un peu plus de 0.5, puis fluctue entre 0.5 et 0.3 tout au

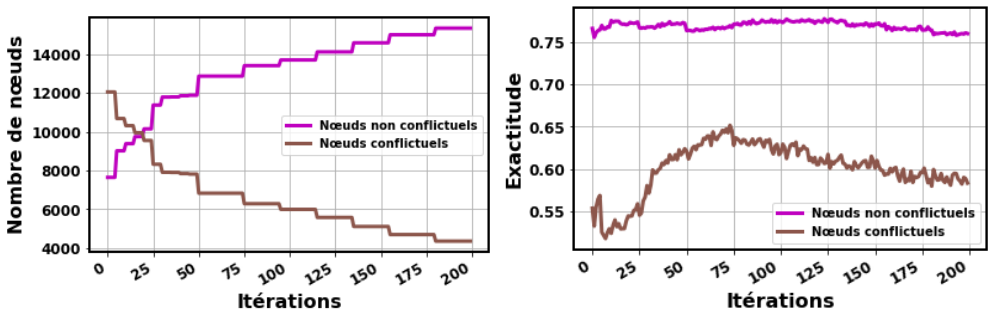
long de l'apprentissage, jusqu'à l'atteinte d'un peu moins de 0.49 à la fin.



(a) (b)

Figure 4.4: Courbes représentatives de l'évolution des nœuds ainsi que leurs exactitudes en utilisant D-DGC avec l'ensemble de données de référence Citeseeer.

Dans la figure 4.5 (a), le nombre de nœuds non conflictuels au départ est de 7651, au contraire des nœuds conflictuels établis à 12066 nœuds. Pendant l'entraînement de D-DGC, le nombre de nœuds non conflictuels augmente périodiquement et constamment dû à la mise à jour des hyperparamètres λ_1 et λ_2 ce qui provoque la baisse du nombre de nœuds conflictuels.



(a) (b)

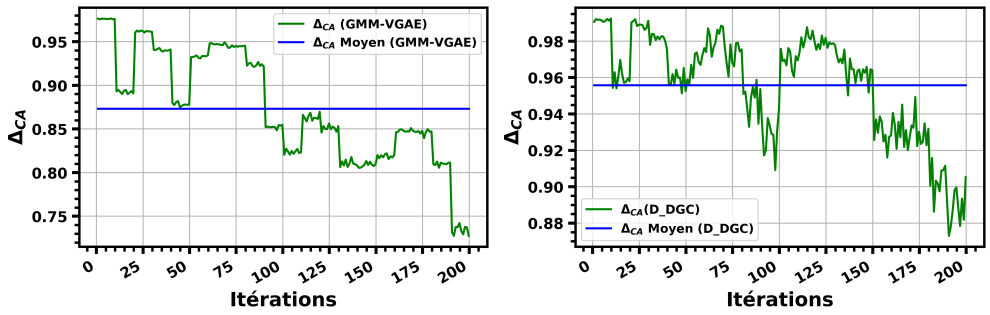
Figure 4.5: Courbes représentatives de l'évolution des nœuds ainsi que leurs exactitudes en utilisant D-DGC avec l'ensemble de données de référence Pubmed.

À l'arrivée des 200 itérations, on retrouve 15361 nœuds non conflictuels bien séparés, alors que 4356 nœuds restent conflictuels. Dans la figure 4.5 (b), l'exactitude des nœuds non conflictuels fluctue à plus de 0.75 tout au long de l'apprentissage de D-DGC. Cependant, l'exactitude des nœuds conflictuels démarre à 0.55, puis fluctue entre 0.5 et 0.65 tout au long de l'apprentissage, pour atteindre à la fin 0.58.

Dans l'ensemble, à chaque itération, le nombre de nœuds en conflit représente la quantité de reconstruction réalisée dans le clustering profond avec D-DGC. Comme on peut le voir sur les figures 4.3, 4.4 et 4.5, le nombre de nœuds en conflit diminue progressivement jusqu'à l'atteinte du nombre minimal à l'approche des 200 itérations de notre modèle (D-DGC). Cependant, le nombre de nœuds non conflictuels connait une augmentation, ce qui est traduit par la bonne identification des clusters par D-DGC. Afin d'éviter l'optimum local, les centroïdes ainsi que λ_1 et λ_2 sont mis à jour automatiquement, d'où le nombre de nœuds en conflit qui diminue progressivement d'une itération à l'autre. Ce résultat confirme que les connaissances acquises en regroupant les nœuds les plus fiables rendent progressivement les nœuds plus difficiles (en conflit) fiables pour le clustering (non conflictuels). En termes d'exactitude, l'apprentissage du modèle démarre avec un taux d'exactitude des nœuds non conflictuels élevé, car ces derniers sont identifiés de manière sure. Au fur et à mesure, celle-ci préserve une constance tout au long de l'apprentissage, au contraire de l'exactitude des nœuds conflictuels qui est plus basse que sa précédente.

Enfin, pour montrer la capacité de notre modèle (D-DGC) à réduire le caractère arbitraire des descripteurs sur les ensembles de données de référence Cora (figure 4.6), Citeseer (figure 4.7), et Pubmed (figure 4.8) ; nous analysons le comportement de la métrique Δ_{CA} pour D-DGC et GMM-VGAE.

La figure 4.6 illustre l'évolution des valeurs de Δ_{CA} pour les modèles D-DGC et GMM-VGAE pendant l'apprentissage sur l'ensemble de données de référence Cora. Au début de l'apprentissage, les valeurs de Δ_{CA} approchent le 0.99 pour GMM-VGAE et 0.97 pour D-DGC. Ensuite, le Δ_{CA} pour GMM-VGAE diminue de manière soudaine alors que pour D-DGC, il se maintient au-dessus de 0.9 pendant 185 itérations. À la fin du processus d'apprentissage, Δ_{CA} pour D-DGC atteint la valeur 0.9, tandis que celui de GMM-VGAE est a 0.81. Cela signifie que la direction du gradient de la fonction de coût dynamique de D-DGC devient très proche de la direction du gradient supervisé (cas idéal où la majorité des étiquettes sont bien prédites). La moyenne de Δ_{CA} durant l'entraînement de D-DGC est de 0.95, alors que celle de GMM-VGAE est de 0.87.

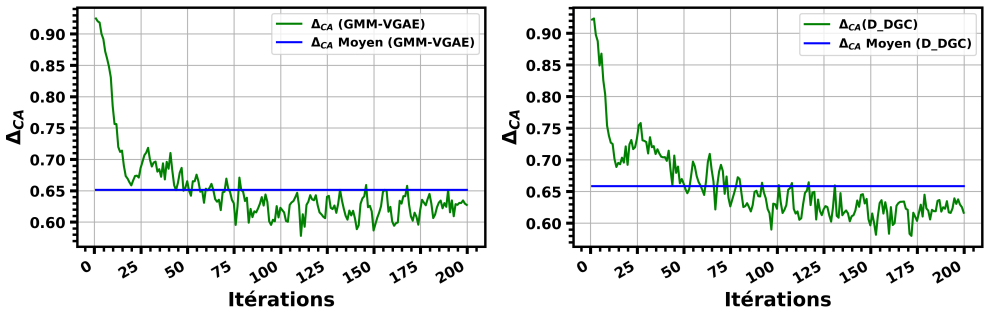


(a) GMM-VGAE (b) D-DGC

Figure 4.6: Courbes représentatives de l'évolution de la métrique Δ_{CA} sur les modèles GMM-VGAE et D-DGC avec l'ensemble de données de référence Cora.

La figure 4.7 illustre l'évolution des valeurs de Δ_{CA} pour les modèles D-DGC et GMM-VGAE pendant l'apprentissage sur l'ensemble de données de référence Citeseer. Au début de l'apprentissage, les valeurs de Δ_{CA} approchent le 0.93 pour GMM-VGAE et D-DGC, ceci indique que le gradient des deux modèles tendent à approximer le gradient supervisé. Au fur et à mesure de l'avancement de l'apprentissage, le Δ_{CA} pour GMM-VGAE et D-DGC suit la mème tendance baissière avec des fluctuations entre 0.75 et 0.55. La moyenne de Δ_{CA} pour l'entraînement

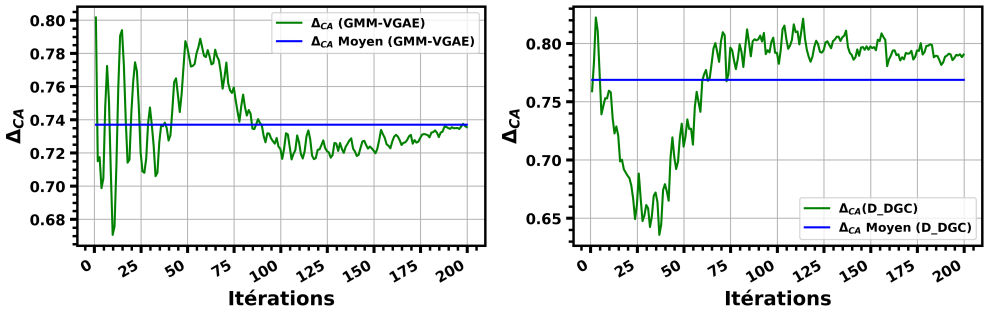
de D-DGC est de 0.66, alors que celle de GMM-VGAE est de 0.64. Dans ce cas, on remarque que pour l'ensemble de données de référence Citeseer, les valeurs moyennes de Δ_{CA} pour les deux modèles sont proches (différence de 0.02), ceci peut-être expliqué par une faible proportion du nombre de nœuds (3312) par rapport au nombre de liens (4732), et du nombre de clusters (6), ce qui limite la capacité des autoencodeurs de graphes à bien incorporer et reconstruire le graphe original.



(a) GMM-VGAE (b) D-DGC

Figure 4.7: Courbes représentatives de l'évolution de la métrique Δ_{CA} sur les modèles GMM-VGAE et D-DGC avec l'ensemble de données de référence Citeseer.

La figure 4.8 illustre l'évolution des valeurs de Δ_{CA} pour les modèles D-DGC



(a) GMM-VGAE (b) D-DGC

Figure 4.8: Courbes représentatives de l'évolution de la métrique Δ_{CA} sur les modèles GMM-VGAE et D-DGC avec l'ensemble de données de référence Pubmed.

et GMM-VGAE pendant l'apprentissage sur l'ensemble de données de référence Pubmed. Au début de l'apprentissage, les valeurs de Δ_{CA} approchent le 0.81 pour GMM-VGAE et 0.76 pour D-DGC. Ensuite, les deux modèles affichent le même comportement. À la fin du processus d'apprentissage, Δ_{CA} pour D-DGC atteint la valeur 0.79 et celui de GMM-VGAE atteint 0.73. La moyenne de Δ_{CA} pour l'entraînement de D-DGC est de 0.77, alors que celle de GMM-VGAE est de 0.73.

Dans l'ensemble, nous pouvons confirmer que notre modèle de clustering profond des graphes attribués est adapté pour atténuer l'effet du caractère arbitraire des descripteurs, ceci est dû notamment à la fonction de coût dynamique du modèle (apprentissage dynamique).

4.4 Conclusion

Les résultats présentés dans ce chapitre pour le modèle D-DGC, sur les ensembles de données de référence du clustering pour les graphes attribués, démontrent que l'approche proposée admet des performances supérieures dans la majorité des cas par rapport aux autres approches existantes. En effet, l'incorporation d'une fonction de coût dynamique permettant de réduire l'impact du caractère arbitraire des descripteurs a permis de former des clusters finaux bien identifiés.

CHAPITRE V

CONCLUSION ET PERSPECTIVES

Dans le cadre de ce mémoire, nous avons exposé la problématique du clustering profond des graphes attribués. Nous avons abordé ce travail en évoquant une revue des principales méthodes décrites dans la littérature, tout en soulignant la problématique du caractère arbitraire des descripteurs, dont souffrent la plupart des solutions de clustering profond des graphes attribués. De ce fait, nous avons proposé une solution basée sur un module d'apprentissage dynamique, afin de réduire graduellement l'effet du caractère arbitraire des descripteurs. La solution proposée (D-DGC) met en place le modèle de clustering profond de graphes attribués, et améliore la fonction de coût de ce dernier en appliquant un apprentissage dynamique permettant de mieux identifier les clusters, et fournissant une performance supérieure aux modèles de l'état de l'art. Indépendamment de son efficacité, l'approche proposée offre une identification des clusters bien plus performante que celle des modèles basés sur l'autoencodeur de graphes. L'évaluation expérimentale confirme la capacité de l'approche à identifier des clusters de manière dynamique, en minimisant l'impact du caractère arbitraire des descripteurs pour l'autoencodeur de graphe.

En terminant, nous estimons que l'approche proposée dans ce travail peut également être reconduite sur d'autres types de graphes, tel que les graphes multiplexes

ou hétérogènes, en prenant en compte les caractéristiques propres à ces types de graphes, afin de réduire l'impact du caractère arbitraire des descripteurs.

RÉFÉRENCES

Balasubramanyan, R. et Cohen, W. W. (2011). Block-lda : Jointly modeling entity-annotated text and entity-entity links. Dans *Proceedings of the 2011 SIAM International Conference on Data Mining*, 450–461.

Baldi, P. (2011). Autoencoders, unsupervised learning and deep architectures. Dans *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop*, UTLW'11, 37—-50.

Bengio, E., Bacon, P.-L., Pineau, J. et Precup, D. (2015). Conditional computation in neural networks for faster models. *arXiv preprint arXiv :1511.06297*.

Bengio, Y. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv : 1308.3432*.

Blei, D. M., Ng, A. Y. et Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, *3*, 993–1022.

Bolukbasi, T., Wang, J., Dekel, O. et Saligrama, V. (2017). Adaptive neural networks for fast test-time prediction. *arXiv preprint arXiv :1702.07811*, *1*(3).

Brunetti, A., Buongiorno, D., Trotta, G. F. et Bevilacqua, V. (2018). Computer vision and deep learning techniques for pedestrian detection and tracking : A survey. *Neurocomputing*, *300*, 17–33.

Cao, S., Lu, W. et Xu, Q. (2016). Deep neural networks for learning graph representations. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, *30*(1), 1145–1152.

Caron, M., Bojanowski, P., Joulin, A. et Douze, M. (2018). Deep clustering for unsupervised learning of visual features. Dans *Proceedings of the European Conference on Computer Vision (ECCV)*, 132–149.

Chen, M., Xu, Z., Weinberger, K. Q. et Sha, F. (2012). Marginalized denoising autoencoders for domain adaptation. Dans *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, 1627—-1634.

Cheng, A.-C., Lin, C. H., Juan, D.-C., Wei, W. et Sun, M. (2020). Instanas : Instance-aware neural architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(04), 3577–3584.

Combe, D., Largeron, C., Géry, M. et Egyed-Zsigmond, E. (2015). I-louvain : An attributed graph clustering method. Dans *International Symposium on Intelligent Data Analysis*, 181–192.

Cruz, J. D., Bothorel, C. et Poulet, F. (2011). Entropy based community detection in augmented social networks. Dans *2011 International Conference on computational aspects of social networks (CASoN)*, 163–168.

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H. et Wei, Y. (2017). Deformable convolutional networks. Dans *Proceedings of the IEEE international conference on computer vision*, 764–773.

Dilokthanakul, N., Mediano, P. A., Garnelo, M., Lee, M. C., Salimbeni, H., Arulkumaran, K. et Shanahan, M. (2016). Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv :1611.02648*.

El Mahrsi, M. K. et Rossi, F. (2012). Graph-based approaches to clustering network-constrained trajectory data. Dans *International Workshop on New Frontiers in Mining Complex Patterns*, 124–137.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P. et Bengio, S. (2010). Why does unsupervised pre-training help deep learning ? *J. Mach. Learn. Res.*, *11*, 625—660.

Firat, A., Chatterjee, S. et Yilmaz, M. (2007). Genetic clustering of social networks using random walks. *Computational Statistics & Data Analysis*, *51*(12), 6285–6294.

Fortunato, S. (2010). Community detection in graphs. *Physics reports*, *486*(3-5), 75–174.

Gao, H., Zhu, X., Lin, S. et Dai, J. (2020). Deformable kernels : Adapting effective receptive fields for object deformation. Dans *8th International Conference on Learning Representations, ICLR 2020*.

Gao, W., Wu, H., Siddiqui, M. K. et Baig, A. Q. (2018). Study of biological networks using graph theory. *Saudi Journal of Biological Sciences*, *25*(6), 1212–1219.

Ge, R., Ester, M., Gao, B. J., Hu, Z., Bhattacharya, B. et Ben-Moshe, B. (2008). Joint cluster analysis of attribute data and relationship data : The connected k-center problem, algorithms and applications. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *2*(2), 1–35.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. et Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144.

Graves, A. (2016). Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv :1603.08983*.

Günnemann, S., Färber, I., Boden, B. et Seidl, T. (2014). Gamer : a synthesis of subspace clustering and dense subgraph mining. *Knowl. Inf. Syst.*, *40*(2), 243–278.

Günnemann, S., Färber, I., Raubach, S. et Seidl, T. (2013). Spectral subspace clustering for graphs with feature vectors. Dans *2013 IEEE 13th International Conference on Data Mining*, 231–240.

Guo, X., Liu, X., Zhu, E. et Yin, J. (2017). Deep clustering with convolutional autoencoders. Dans *International conference on neural information processing*, 373–382.

Han, Y., Huang, G., Song, S., Yang, L., Wang, H. et Wang, Y. (2021). Dynamic neural networks : A survey. *arXiv preprint arXiv :2102.04906*.

Heaney, M. T. (2014). Multiplex networks and interest group influence reputation : An exponential random graph model. *Social Networks*, *36*, 66–81.

Huang, G., Chen, D., Li, T., Wu, F., Van Der Maaten, L. et Weinberger, K. Q. (2017). Multi-scale dense convolutional networks for efficient prediction. *arXiv preprint arXiv :1703.09844*, *2*.

Huang, J. et Ling, C. X. (2005). Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering*, *17*(3), 299–310.

Huang, P., Huang, Y., Wang, W. et Wang, L. (2014). Deep embedding network for clustering. Dans *2014 22nd International conference on pattern recognition*, 1532–1537.

Hubert, L. et Arabie, P. (1985). Comparing partitions. *Journal of classification*, *2*(1), 193–218.

Hui, B., Zhu, P. et Hu, Q. (2020). Collaborative graph convolutional networks : Unsupervised learning meets semi-supervised learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(04), 4215–4222.

Izhikevich, E. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, *14*(6), 1569–1572.

Jiang, Z., Zheng, Y., Tan, H., Tang, B. et Zhou, H. (2017). Variational deep embedding : An unsupervised and generative approach to clustering. Dans C. Sierra (dir.). *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*, 1965–1972.

Kingma, D. P. et Welling, M. (2014). Auto-encoding variational bayes. Dans Y. Bengio et Y. LeCun (dir.). *2nd International Conference on Learning Representations, ICLR 2014*.

Kipf, T. N. et Welling, M. (2016). Variational graph auto-encoders. *arXiv preprint arXiv :1611.07308*.

Kipf, T. N. et Welling, M. (2017). Semi-supervised classification with graph convolutional networks. Dans *5th International Conference on Learning Representations, ICLR 2017*.

Lee, D. et Shim, J. (2015). Impact parameter analysis of subspace clustering. *International Journal of Distributed Sensor Networks*, *11*(9), 398–452.

Li, Stan Z.and Jain, A. (2009). L2 norm. Dans *Encyclopedia of Biometrics*, 883–883.

Li, Y., Luo, C. et Chung, S. M. (2008). Text clustering with feature selection by using statistical data. *IEEE Transactions on knowledge and Data Engineering*, *20*(5), 641–652.

Liu, Y., Niculescu-Mizil, A. et Gryc, W. (2009). Topic-link lda : joint models of topic and author community. Dans *proceedings of the 26th annual international conference on machine learning*, 665–672.

Mogotsi, I. (2010). Christopher d. manning, prabhakar raghavan, and hinrich schütze : Introduction to information retrieval. *Information Retrieval*, *13*(2), 192–195.

Mrabah, N., Khan, N. M., Ksantini, R. et Lachiri, Z. (2020). Deep clustering with a dynamic autoencoder : From reconstruction towards centroids construction. *Neural Networks*, *130*, 206–228.

Ng, A. Y., Jordan, M. I. et Weiss, Y. (2002). On spectral clustering : Analysis and an algorithm. Dans *Advances in neural information processing systems*, 849–856.

Niwa, Y. et Nitta, Y. (1994). Co-occurrence vectors from corpora vs. distance vectors from dictionaries. *Proceedings of the 15th Conference on Computational Linguistics*, *1*, 304—-309.

Noh, J. D. et Rieger, H. (2004). Random walks on complex networks. *Physical review letters*, *92*(11), 118701.

Olteanu, M., Villa-Vialaneix, N. et Cierco-Ayrolles, C. (2013). Multiple kernel self-organizing maps. Dans *21. European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2013)*.

Palshikar, G. K. et Apte, M. M. (2008). Collusion set detection using graph clustering. *Data mining and knowledge Discovery*, *16*(2), 135–164.

Pan, L., Dai, X., Huang, S. et Chen, J. (2015). Academic paper recommendation based on heterogeneous graph. Dans *Chinese computational linguistics and natural language processing based on naturally annotated big data*, 381–392.

Pan, S., Hu, R., Long, G., Jiang, J., Yao, L. et Zhang, C. (2018). Adversarially regularized graph autoencoder for graph embedding. Dans *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, 2609–2615.

Perozzi, B., Al-Rfou, R. et Skiena, S. (2014). Deepwalk : Online learning of social representations. Dans *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710.

Reynolds, D. (2009). Gaussian mixture models. Dans S. Z. Li et A. Jain (dir.). *Encyclopedia of Biometrics*, 659–663.

Rossi, R. et Ahmed, N. (2015). The network data repository with interactive graph analytics and visualization. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, *29*(1), 4292–4293.

Shen, J., Wang, Y., Xu, P., Fu, Y., Wang, Z. et Lin, Y. (2020). Fractional skipping : Towards finer-grained dynamic cnn inference. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(04), 5700–5708.

Singh, K., Shakya, H. K. et Biswas, B. (2016). Clustering of people in social network based on textual similarity. *Perspectives in Science*, *8*, 570–573.

Song, S., Zhu, Y. et Yu, H. (2005). A power network topology tracking method based on graph theory and artificial intelligence search technique. *Power System Technology*, *29*(19), 45–49.

Suetin, P., Kostrikin, A. I. et Manin, Y. I. (1997). *Linear Algebra and Geometry*, volume 1.

Talwar, P., Silla, Y., Grover, S., Gupta, M., Agarwal, R., Kushwaha, S. et Kukreti, R. (2014). Genomic convergence and network analysis approach to identify candidate genes in alzheimer's disease. *BMC genomics*, *15*(1), 1–16.

Tang, J., Aggarwal, C. et Liu, H. (2016). Node classification in signed social networks. Dans *Proceedings of the 2016 SIAM international conference on data mining*, 54–62.

Teerapittayanon, S., McDanel, B. et Kung, H.-T. (2016). Branchynet : Fast inference via early exiting from deep neural networks. Dans *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2464–2469.

Tong, C. (2019). Statistical inference enables bad science ; statistical thinking enables good science. *The American Statistician*, *73*(sup1), 246–261.

Wang, C., Pan, S., Hu, R., Long, G., Jiang, J. et Zhang, C. (2019). Attributed graph clustering : A deep attentional embedding approach. Dans *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 3670–3676.

Wang, C., Pan, S., Long, G., Zhu, X. et Jiang, J. (2017a). Mgae : Marginalized graph autoencoder for graph clustering. Dans *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 889–898.

Wang, D., Cui, P. et Zhu, W. (2016). Structural deep network embedding. Dans *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 1225–1234.

Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W. et Yang, S. (2017b). Community preserving network embedding. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, *31*(1), 203–209.

Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L. S., Grauman, K. et Feris, R. (2018). Blockdrop : Dynamic inference paths in residual networks. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8817–8826.

Xie, J., Girshick, R. et Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, *48*, 478—-487.

Xu, Z., Ke, Y., Wang, Y., Cheng, H. et Cheng, J. (2012). A model-based approach to attributed graph clustering. Dans *Proceedings of the 2012 ACM SIGMOD international conference on management of data*, 505–516.

Yang, B., Bender, G., Le, Q. V. et Ngiam, J. (2019). Condconv : Conditionally parameterized convolutions for efficient inference. *Advances in Neural Information Processing Systems*, *32*, 1305–1316.

Yang, B., Fu, X., Sidiropoulos, N. D. et Hong, M. (2017). Towards k-means-friendly spaces : Simultaneous deep learning and clustering. *Proceedings of the 34th International Conference on Machine Learning*, *70*, 3861–3870.

Yao, B., Liu, X., Zhang, W.-J., Chen, X.-E., Zhang, X.-M., Yao, M. et Zhao, Z.-X. (2013). Applying graph theory to the internet of things. Dans *2013 IEEE 10th International Conference on High Performance Computing and Communications 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, 2354–2361.

Young, T., Hazarika, D., Poria, S. et Cambria, E. (2018). Recent trends in deep learning based natural language processing. *ieee Computational intelligence magazine*, *13*(3), 55–75.

Zhou, Y., Cheng, H. et Yu, J. X. (2009). Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, *2*(1), 718–729.

Zhu, X., Hu, H., Lin, S. et Dai, J. (2019). Deformable convnets v2 : More deformable, better results. Dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9308–9316.