DspGNN : Une Approche Spectrale de Réseau de Neurones sur Graphes Dynamiques pour la régression des arêtes

Leshanshui Yang*,**, Clément Chatelain***, Sébastien Adam**

*Saagie, 72 Rue de la République, 76140 Le Petit-Quevilly, France
leshanshui.yang@saagie.com,
https://www.saagie.com/fr/
**Univ Rouen Normandie, LITIS UR 4108, F-76000 Rouen, France
sebastien.adam@univ-rouen.fr
***INSA Rouen Normandie, LITIS UR 4108, F-76000 Rouen, France
clement.chatelain@insa-rouen.fr

Résumé. Cet article présente DspGNN (Dynamic Spectral-Parsing Graph Neural Network), un réseau de neurones sur graphes dynamiques qui intègre des opérations de convolution spectrales sur graphes. DspGNN permet de capturer efficacement les informations spectrales évolutives dans des graphes dynamiques à temps discret (DTDG), à des fins de prédiction d'attributs numériques sur les arêtes. Notre première contribution est l'adaptation et l'optimisation des convolutions spectrales sur graphes dynamiques qui étaient jusqu'ici dédiées aux graphes statiques. La seconde contribution est une technique innovante, simple et efficace appelée Active Node Mapping (ANM) pour réduire la complexité calculatoire de la décomposition en valeurs propres sur de grands DTDGs. Au travers d'expérimentations, nous montrons que le modèle DspGNN est performant sur l'ensemble de données de transactions en bitcoins de la conférence EGC 2024 et sur deux autres ensembles de données de la littérature.

1 Introduction

Les graphes sont utilisés pour modéliser des systèmes complexes dans des applications du monde réel pour leur capacité à représenter les données structurées. Dans les cas où la structure ou les attributs évoluent avec le temps, on utilise des graphes dits dynamiques, comme c'est le cas dans les réseaux de transactions (Zhou et al. (2023)). Dans ces graphes dynamiques, les nœuds représentent les utilisateurs et les arêtes modélisent des relations entre les nœuds au cours du temps. Des attributs numériques sont parfois associés à ces arêtes, tels que des volumes de transactions ou des scores d'évaluation (Kumar et al. (2016); Zhou et al. (2023)). On distingue les graphes dynamiques à temps discret (DTDGs) des graphes dynamiques à temps continu (CTDGs). Les DTDGs utilisent des séquences de graphes statiques (on parlera dans cet article de "snapshot") pour représenter le graphe dynamique, tandis que les CTDGs utilisent des arêtes horodatées. Si certaines études (Jiang et al. (2023); Raghavendra et al. (2022)) se sont intéressées à la régression d'arête sur les CTDGs, ce n'est pas le cas pour des

DTDGs, alors que ce problème revet un potentiel applicatif important, en particulier en ce qui concerne la prédiction des transactions en bitcoins (Zhou et al. (2023)).

Cet article aborde le problème de la régression d'arêtes dans des graphes dynamiques à temps discret (DTDGs). Nous y présentons un nouveau modèle nommé DspGNN (Dynamic Spectral-Parsing Graph Neural Network) qui est, à notre connaissance, la première application aux DTDGs de réseau de convolution sur graphes conçu dans le domaine spectral. DspGNN utilise un banc de filtres, là où les modèles classiques se limitent généralement aux filtres passe-bas (Balcilar et al. (2020)). Pour effectuer une décomposition spectrale efficace sur des DTDGs à large échelle, nous introduisons aussi une méthode appelée Active Node Mapping (ANM) qui réduit fortement le coût calculatoire. À des fins de comparaison de performances, nous proposons également une modification des modèles de référence (Pareja et al. (2020); Wang et al. (2023)), connus pour leurs performances en classification des arêtes et en régression des nœuds sur DTDGs, mais qui ne sont pas adaptés à une tâche de régression des attributs des arêtes. Une évaluation comparative sur trois ensembles de données - EGC 2024, Bitcoin-Alpha et Bitcoin-OTC (Kumar et al. (2016)) - montre non seulement l'efficacité de notre technique en terme de temps de calcul, mais aussi une amélioration significative des performances par rapport aux modèles de référence, démontrant son potentiel, notamment dans le contexte des réseaux de transactions.

2 Contexte et travaux connexes

La convolution sur graphes est un concept fondamental au sein des réseaux de neurones sur graphes. On distingue généralement deux approches principales : les convolutions spatiales et les convolutions spectrales (Balcilar et al. (2020)).

Les convolutions spatiales sur graphes agrégent les informations locales des voisins. Une couche de convolution spatiale réalise le calcul $\mathbf{H}' = \sigma\left(\sum_{\mathbf{s}} \mathbf{C}^{(\mathbf{s})}\mathbf{H}\mathbf{W}^{(\mathbf{s})}\right)$ (Balcilar et al. (2020)), où $\mathbf{H}$ et $\mathbf{H}'$ représentent les matrices d'états cachés des nœuds en entrée et en sortie, σ est une fonction d'activation, les $\mathbf{C}^{(\mathbf{s})}$ sont les noyaux de convolution (par exemple, la matrice d'adjacence $\mathbf{A}$) et $\mathbf{W}$ est la matrice de paramètres entraînables qui projette les états cachés de dimension $d_{\mathbf{H}}$ dans une dimension $d_{\mathbf{H}'}$.

Les convolutions spectrales sur graphes s'appuient quant à elles sur la théorie spectrale des graphes. Elles reposent sur une décomposition de la matrice laplacienne (normalisée) $\mathbf{L}^{(\text{norm})} = \mathbf{D}^{-\frac{\mathbf{1}}{\mathbf{2}}}\mathbf{A}\mathbf{D}^{-\frac{\mathbf{1}}{\mathbf{2}}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathbf{T}}$, où $\mathbf{A}$ et $\mathbf{D}$ représentent respectivement la matrice d'adjacence et la matrice diagonale des degrés des nœuds. Grâce à cette décomposition, la Transformée de Fourier du graphe (Graph FT) peut être calculée par le produit du signal par la matrice des vecteurs propres $\mathbf{U}^{\mathbf{T}}$ de $\mathbf{L}$. Divers filtres peuvent ensuite être appliqués, et ces signaux filtrés peuvent être ramenés dans le domaine spatial par la Transformée de Fourier Inverse du graphe (Graph IFT) avec $\mathbf{U}$. Une expression générale est présentée comme $\mathbf{H}' = \mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^{\mathbf{T}}\mathbf{H}$, où $\mathbf{\Lambda}$ est une matrice diagonale des valeurs propres et $g(\cdot)$ est une fonction de filtre qui contrôle la réponse en fréquence.

Des résultats récents ont montré que les GNNs spatiaux agissent comme des filtres passe-bas dans le domaine spectral, et qu'en ajoutant des filtres passe-haut et/ou passe-bande, les performances peuvent être améliorées (Balcilar et al. (2020)). Ainsi, le modèle Spectral-Designed Graph Convolution Network (DSGCN) propose de calculer des noyaux de convolution en filtrant les valeurs propres de la matrice laplacienne, indiqué comme $\mathbf{H}' = \sigma\left(\sum_{\mathbf{s}} \mathbf{C}^{(\mathbf{s})}\mathbf{H}\mathbf{W}^{(\mathbf{s})}\right)$,

$\mathbf{C^s} = \mathbf{U\Phi^s}(\mathbf{\Lambda})\mathbf{U^T}$. Ces noyaux de convolution peuvent être considérés comme des matrices d'adjacence pondérées pour la convolution spatiale, mais conçues du point de vue du filtrage dans le domaine spectral. DSGCN fait le lien entre la convolution spectrale et spatiale, tout en proposant des performances supérieures aux GNNs de la littérature.

Réseaux de neurones sur les graphes dynamiques en temps discret Un DTDG est représenté par une séquence de graphes statiques $(G_1, G_2, ..., G_T)$. Chaque G_t est représenté par sa matrice d'adjacence $\mathbf{A}_t$, éventuellement complétée par $\mathbf{X}_t^{node}$ et/ou $\mathbf{X}_t^{edge}$ qui représentent respectivement les attributs des nœuds ou des arêtes. Les tâches dans les DTDGs impliquent généralement de prédire les attributs ou la connectivité des graphes futurs basés sur K snapshots précédents (Pareja et al. (2020); Wang et al. (2023)). Un avantage des Réseaux de Neurones sur les DTDGs (DTDGNNs) est leur capacité à encoder chaque G_t en utilisant les modèles de graphes statiques. Ces modèles emploient un GNN f_G pour encoder les états cachés de chaque graphe, puis propagent l'information entre K graphes avec un module temporel f_T tel que les Réseaux de Neurones Récurrents (RNNs) (voir l'Eq. 1). CoEvoGNN (Wang et al. (2023)) suit ce paradigme utilisant GraphSage (Hamilton et al. (2017)) en tant que f_G et la somme normalisée L2 comme f_T pour la dimension temporelle. Ce modèle a contribué à l'application des DTDGNNs dans la tâche de régression de nœuds sur les réseaux de transaction.

$$\mathbf{H'}_k = f_G(\mathbf{A}_k, \mathbf{H}_k), \quad k \in [t-K, t-1], \quad \mathbf{H}_t = f_T(\mathbf{H'}_{t-K:t-1}) \tag{1}$$

En plus de l'encodage séquentiel des états cachés, une autre catégorie vise à convoluer différemment les graphes à chaque pas de temps pour mieux capturer les dynamiques. Un exemple typique est EvolveGCN (Pareja et al. (2020)), qui met à jour les paramètres de poids d'un GCN $\mathbf{\Theta}_{f_G,t} = f_T(\mathbf{\Theta}_{f_G,t-1}, \mathbf{H}_{t-1})$ pour chaque graphe via un RNN. Néanmoins, à notre connaissance, la conception de l'architecture et l'analyse des performances des méthodes à conception spectrale sur les DTDGs sont encore largement inexplorées.

3 Convolution spectrale sur graphes dynamiques

Certaines études récentes telles que (Huang et al. (2023)) ont montré que l'évolution du spectre à chaque pas de temps permet de caractériser des DGs, avec des applications à la détection d'anomalies sur des DGs. En combinant les concepts clés décrits dans la section 2, nous proposons de remplacer le module de convolution des DTDGNNs par un GNN spectral, afin d'obtenir un encodeur spectral de graphe dynamique. La conception de ce modèle est décrite dans cette section : la section 3.1 présente notre modèle appelé **DspGNN (Dynamic Spectral-Parsing Graph Neural Network)**, qui exploite les avantages des GNNs conçus spectralement. Dans la section 3.2, nous présentons une technique originale appelée **ANM (Active Node Mapping)** pour surmonter le défi combinatoire de la décomposition spectrale sur les DTDGs.

3.1 Analyse spectrale sur graphes : du statique au dynamique

De manière classique, notre modèle prend en entrée les matrices d'adjacence $\mathbf{A}_{t-K:t-1} \in \mathbb{R}^{K\times N\times N}$ et les états cachés $\mathbf{H}_{t-K:t-1} \in \mathbb{R}^{K\times N\times d}$ des K snapshots précédents pour calculer

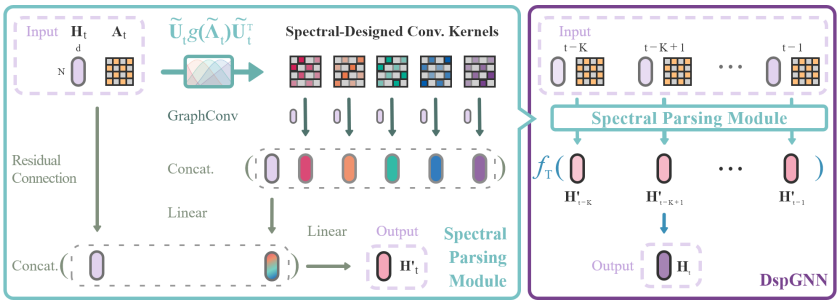

FIG. 1 – *Gauche : Notre GNN de conception spectrale, où différents noyaux de convolutions sont obtenus à travers des filtres conçus spectralement. Les informations propagées sont ensuite fusionnées avec des poids apprenables, concaténées sur* $\boldsymbol{H}$*, et transmises dans une couche linéaire. Droite : L'architecture globale de DspGNN implique l'encodage des états cachés des* K *snapshots passés par sp-GNN. Cette séquence d'états cachés est ensuite encodée par un module temporel, pour obtenir l'état caché pour le pas de temps actuel.*

l'état caché courant $\mathbf{H}_t \in \mathbb{R}^{N \times d}$ par un DTDGNN. Une couche de décision effectue ensuite la tâche de régression, produisant $\hat{\mathbf{Y}} \in \mathbb{R}^{N \times d_{\text{pred}}}$.

Comme dans DSGCN (Balcilar et al. (2020)), nous appliquons d'abord une décomposition en valeurs propres du laplacien normalisé $\mathbf{L}^{\text{norm}}$ à chaque pas de temps. Les valeurs propres sont filtrées par S filtres spectraux distincts qui sont inversés pour produire des noyaux de convolution, formalisé comme $\mathbf{C}^s = \mathbf{U} g^s(\mathbf{\Lambda}) \mathbf{U}^{\mathbf{T}} \quad s = 1, 2, \ldots, S$. Notre modèle considère un banc de filtre comprenant un filtre passe-bas $F_{bas}(\lambda) = (1 - \lambda/\lambda_{\text{max}})^3$, un filtre passe-haut $F_{haut}(\lambda) = \frac{\lambda}{\lambda_{\text{max}}}$, et trois filtres passe-bande $F_{bande_i}(\lambda) = e^{-\gamma(\lambda - \lambda_{c_i})^2}$ de paramètres ajustables λ_{c_i} (fréquences centrales) et γ (largeur de bande) comme dans DSGCN.

Notre amélioration du modèle DSGCN repose sur l'apprentissage de la combinaison de plusieurs noyaux de convolutions, éliminant ainsi le besoin de sélection manuelle de filtres à travers différents jeux de données. Pour ce faire, nous introduisons une couche linéaire avec une fonction d'activation pour apprendre la combinaison non linéaire des sorties de chaque filtre et ainsi réduire la dimension de $d \times S$ à d. Nous donnons au modèle la capacité d'apprendre l'importance de chaque filtre pendant la phase d'entraînement, automatisant ainsi le processus de sélection et de pondération des différents filtres.

Notre approche **Spectral-Parsing GNN** permet donc non seulement d'analyser les graphes dynamiques dans le domaine spectral, mais aussi de sélectionner et de combiner efficacement les différentes bandes de fréquences (voir figure 1 (à gauche)). L'architecture globale de DspGNN encode séquentiellement les états cachés, avec une analyse spectrale appliquée à chaque pas de temps, puis encode l'ensemble de la séquence par un réseau de neurones LSTM (voir équations 1 et figure 1 (à droite)).

3.2 Noyaux de convolution conçus spectralement sur les DTDGs

Le traitement de graphes dynamiques pose deux défis majeurs pour le calcul de noyaux de convolution dans le domaine spectral. Le premier est la complexité calculatoire de la décom-

position spectrale, avec une complexité en temps de $O(N^3)$ par pas de temps, où N représente le nombre total de nœuds [1]. Le second défi est la variabilité dans la distribution des valeurs propres en fonction de la position dans la séquence.

Pour aborder le premier problème, nous proposons une technique simple et efficace baptisée **Active Node Mapping (ANM)**. Cette technique représente chaque pas de temps avec ses nœuds actifs par la matrice d'adjacence $\tilde{\mathbf{A}}_t$, réduisant ainsi la taille de $N \times N$ à $N_t \times N_t$, où N_t représente le nombre de nœuds ayant des liens au moment t, comme illustré par la figure. 2. La méthode ANM améliore considérablement la vitesse de calcul et réduit l'utilisation de la RAM, notamment quand le nombre de nœuds actifs N_t est faible par rapport à N. Les résultats d'accélération sont présentés dans le Tableau 3. Nous proposons également de rendre les filtres passe-bande adaptatifs, en les centrant sur les percentiles $\frac{1}{nbf+1}, \frac{2}{nbf+1}, \ldots, \frac{nbf}{nbf+1}$ des valeurs propres, où nbf représente le nombre de filtres, permettant ainsi une analyse spectrale qui s'ajuste à l'évolution des snapshots.

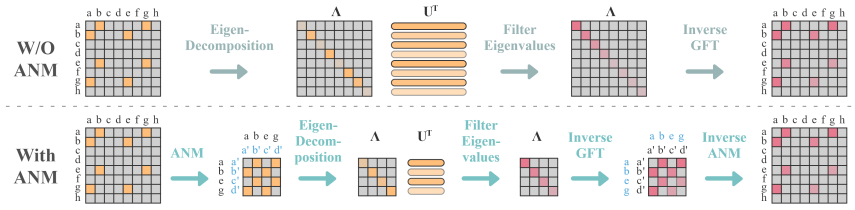

FIG. 2 – *haut : Décomposition en valeurs propres sur la matrice d'adjacence complète avec N nœuds. Bas : réduction de la complexité temporelle de $O(N^3)$ à $O(N_t^3)$, avec N_t nœuds actifs au lieu de l'ensemble des N nœuds. Active Node Mapping réduit la consommation de temps à $(\frac{N_t}{N})^3$.*

4 Évaluation expérimentale

4.1 Jeux de données

Dans cette section, nous présentons les expériences menées pour valider expérimentalement le modèle présenté dans la section précédente. Ces expériences sont réalisées sur l'ensemble de données du "défi EGC 2024" [2], mais aussi sur deux autres jeux de données accessibles au public : Bitcoin-Alpha (BCA) et Bitcoin-OTC (BCO) (Kumar et al. (2016)). La tâche consiste à prédire les attributs des arêtes d'un snapshot de graphe dynamique, en utilisant comme entrée les matrices d'adjacence des graphes précédents.

Les statistiques concernant les trois jeux de données sont présentées dans le Tableau 1. Dans les jeux de données BCA et BCO, l'attribut d'arête représente une note donnée par un utilisateur à un autre, variant de -10 à 10. Dans le jeu de données EGC, il y a deux attributs, représentant respectivement le nombre de transactions et le montant total des transactions entre

1. Des optimisations pour la décomposition en valeurs propres sur des matrices creuses existent, telles que la méthode de Lanczos qui peut réduire la complexité à $O(dN^2)$, où d est le nombre moyen d'éléments non nuls dans une rangée. Mais ces approches ne cassent pas la complexité liée au grand nombre de nœuds N.
2. https ://iutdijon.u-bourgogne.fr/egc2024/defi-egc/

deux utilisateurs au sein d'une journée, avec des plages allant respectivement de 1×10^0 à 3×10^4 et de 1×10^1 à 7×10^{14}.

4.2 Protocole expérimental

Pour ces tâches de régression, nous normalisons la vraie valeur y_{vrai} entre 0 et 1. Une normalisation Min-Max est utilisée pour les jeux de données BCA et BCO. Étant donné que les attributs du "défi EGC" sont d'un ordre de grandeur plus important, nous avons d'abord appliqué une transformation logarithmique $\log_{10}(y + 1)$ avant la normalisation Min-Max. Conformément au protocole de régression des nœuds dans (Wang et al. (2023)), les mesures d'évaluation sont RMSE (Root Mean Squared Error) et MAE (Mean Absolute Error). Le score final est la moyenne des scores pour chaque pas de temps testé.

Afin de maintenir la cohérence du protocole expérimental pour les jeux de données BCA et BCO, nous suivons exactement la construction et la division des snapshots proposée dans les articles décrivant EvolveGCN et CoEvoGNN, en utilisant les premiers 70% des snapshots pour l'apprentissage, les 10% suivants pour la validation et les 20% restants pour les tests, comme indiqué dans le Tableau 1. Pour que le jeu de données d'EGC ait des snapshots du même ordre de grandeur que les autres datasets, nous avons l'agrégé par tranches de 7 jours. [3]

TAB. 1 – *Informations statistiques sur les jeux de données, où les données d'EGC sont agrégés par périodes de 7 jours.*

Jeux de données	# Nœuds	# Moyenne des nœuds actifs par snapshot	# Arêtes	# Snapshots	Division de snapshots (Appr. / Valid. / Test)
Bitcoin-Alpha	3,783	106	24,173	137	95 / 14 / 28
Bitcoin-OTC	5,881	148	35,588	136	95 / 13 / 28
EGC (agrégé)	9,975	740	2,779,244	131	92 / 13 / 26

4.3 Résultats

Pour évaluer la variance des résultats et garantir la reproductibilité des expériences, chaque expérience est exécutée cinq fois avec des graines aléatoires différentes mais fixes (0, 1, 2, 3, 4). Ces graines sont utilisées pour initialiser les paramètres du modèle ainsi que les états cachés des nœuds. Pour garantir l'équité dans les expériences, nous utilisons les hyperparamètres par défaut de CoEvoGNN, marqués comme CoEvoGNN* dans le Tableau 2, et appliquons les mêmes hyperparamètres optimisés pour CoEvoGNN, EvolveGCN et DspGNN pour que tous les modèles convergents bien.

Les résultats obtenus lors des cinq essais avec différentes graines aléatoires sont présentés dans le tableau 2. Pour les trois ensembles de données, DspGNN surpasse les modèles de base CoEvoGNN et EvolveGCN. Ces résultats confirment l'idée qu'une approche conçue dans le domaine spectral peut surpasser les méthodes uniquement basées sur la convolution spatiale. De plus, malgré la variabilité dans l'initialisation des paramètres et des états cachés, DspGNN

3. Par exemple, si le nœud 'a' effectue un paiement au nœud 'b' le 3ème, le 6ème et le 12ème jour, alors il y a un lien dans le snapshot 1 (jour 1 à jour 7) où 'a' a effectué deux paiements à 'b', et dans le snapshot 2 (jour 8 à jour 14), il y a un lien où 'a' a fait un paiement à 'b'.

TAB. 2 – *Moyenne et écart-type des scores normalisés pour cinq essais de régression des arêtes. Les meilleurs résultats apparaissent en gras.*

	Bitcoin-Alpha		Bitcoin-OTC		EGC (agrégé)	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
CoEvoGNN*	0.1171 (0.01)	0.0879 (0.01)	0.1638 (0.01)	0.1145 (0.01)	0.1153 (0.01)	0.0854 (0.00)
CoEvoGNN	0.1231 (0.00)	0.0928 (0.00)	0.1671 (0.01)	0.1176 (0.01)	0.1115 (0.00)	0.0833 (0.00)
EvolveGCN	0.1173 (0.02)	0.0903 (0.02)	0.1556 (0.01)	0.1025 (0.01)	0.1199 (0.01)	0.0911 (0.01)
DspGNN	**0.0968** (0.00)	**0.0637** (0.00)	**0.1471** (0.00)	**0.0831** (0.00)	**0.1043** (0.00)	**0.0758** (0.00)

présente des écart-types plus faibles que les modèles de références. Ce faible écart-type s'explique selon nous par la couche linéaire entraînable qui combine les représentations filtrées dans le Spectral-Parsing GNN.

Concernant l'accélération de la décomposition spectrale, nous fournissons une analyse théorique et empirique du coût computationnel pour la décomposition spectrale avec et sans Active Node Mapping (ANM). La complexité spatiale maximale théorique sans ANM est $O(N^2)$ et la complexité temporelle est $O(N^3)$ (multiplié par T, le nombre de snapshots). En contraste, avec ANM, la complexité spatiale maximale est déterminée par le plus grand nombre de nœuds actifs $O(\max(N_t)^2)$, et la complexité temporelle est réduite à la somme des cubes du nombre de nœuds actifs par snapshot, $\Sigma_t(N_t)^3$. Pour l'évaluation empirique, nous avons effectué cinq essais de décomposition spectrale pour tous les snapshots avec des configurations constantes, sans inclure le temps Entrée/Sortie. Le tableau 3 présente les temps médians pour chaque approche.

TAB. 3 – *Comparaison de coût calculatoire avec et sans Active Node Mapping (ANM). La « complexité spatiale maximale » fait référence à la complexité spatiale théoriquement requise. « Temps consommé » indique le temps pour effectuer une décomposition spectrale sur tous les snapshots. W/O ANM utilise un algorithme numpy standard ou une version optimisée pour les matrices creuses de scipy , et ANM utilise uniquement l'algorithme numpy standard.*

		Sans Active Node Mapping	Avec Active Node Mapping
Complexité spatiale maximale (Théo.)	Bitcoin-Alpha	1×10^7	3×10^5
	Bitcoin-OTC	3×10^7	3×10^5
	MovieLens-100K	1×10^8	6×10^6
Temps consommé (seconds)	Librairie	Numpy	Scipy	Numpy
	Bitcoin-Alpha	1826.31	225.69	0.31
	Bitcoin-OTC	7203.62	592.23	0.55
	MovieLens-100K	11150.19	1159.91	12.83

5 Conclusion

Dans ce travail, nous avons présenté le Dynamic Spectral-Parsing Graph Neural Network (DspGNN), un modèle inpiré des idées de DSGCN (Balcilar et al. (2020)) et de l'architecture de CoEvoGNN (Wang et al. (2023)) pour la tâche de régression. Notre approche présente trois grandes nouveautés : 1) elle adapte les méthodes spectrales pour une meilleure compatibilité avec les DTDGs 2) elle améliore le modèle DSGCN existant en combinant les informations de

différents noyaux avec un module apprenant, et 3) elle résout de manière innovante le défi de la décomposition spectrale sur les DTDGs avec l'Active Node Mapping (ANM).

Remerciements : Cette recherche a été soutenu financièrement par l'ANR Labcom Lisa (ANR-20-LCV1-0009).

Références

Balcilar, M., G. Renton, P. Héroux, B. Gauzere, S. Adam, et P. Honeine (2020). Bridging the gap between spectral and spatial domains in graph neural networks.

Hamilton, W., Z. Ying, et J. Leskovec (2017). Inductive representation learning on large graphs.

Huang, S., J. Danovitch, G. Rabusseau, et R. Rabbany (2023). Fast and attributed change detection on dynamic graphs with density of states. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 15–26. Springer.

Jiang, L., C. Zhang, F. Poursafaei, et S. Huang (2023). Towards temporal edge regression : A case study on agriculture trade between nations.

Kumar, S., F. Spezzano, V. Subrahmanian, et C. Faloutsos (2016). Edge weight prediction in weighted signed networks. In *International Conference on Data Mining*, pp. 221–230.

Pareja, A., G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, et C. Leiserson (2020). Evolvegcn : Evolving graph convolutional networks for dynamic graphs. In *AAAI conference on artificial intelligence*, Volume 34, pp. 5363–5370.

Raghavendra, M., K. Sharma, S. Kumar, et al. (2022). Signed link representation in continuous-time dynamic signed networks.

Wang, D., Z. Zhang, Y. Ma, T. Zhao, T. Jiang, N. V. Chawla, et M. Jiang (2023). Modeling co-evolution of attributed and structural information in graph sequence. *IEEE Transactions on Knowledge and Data Engineering 35*(2), 1817–1830.

Zhou, Y., X. Luo, et M. Zhou (2023). Cryptocurrency transaction network embedding from static and dynamic perspectives : An overview. *IEEE/CAA Journal of Automatica Sinica 10*(5), 1105–1121.

Summary

We introduce the Dynamic Spectral-Parsing Graph Neural Network (DspGNN), a novel model that incorporates spectral-designed graph convolution for representation learning and edge regression on Discrete Time Dynamic Graphs (DTDGs). Our first major contribution is the optimization of spectral-designed methods to better capture evolving spectral information on DTDGs. Secondly, to solve the computational challenge of performing eigendecomposition on large DTDGs, we propose a novel technique, Active Node Mapping, that proves to be both simple and effective. Our model consistently outperforms baseline methods on three publicly available datasets for edge regression tasks.