Élagage de Graphes sur Terminaux pour un Apprentissage Fédéré Efficace en Communication

(Edge-GNP: On-Device Graph Pruning for Communication-Efficient Federated Learning)

Joshua Juste Emmanuel Yun Pei NIKIEMA
Professeur: Professeur émerite Michel HABIB
Algorithmics, Complexity, and Graph Algorithms

15 février 2026

Résumé

Ce rapport présente une approche novatrice combinant l'apprentissage fédéré (Federated Learning) et l'élagage de graphes pour l'entraînement distribué de réseaux de neurones graphiques (GNN) sur des terminaux à ressources limitées. Le projet *Edge-GNP* (On-Device Graph Pruning for Communication-Efficient FL) vise à réduire les coûts de communication tout en préservant la performance des modèles en exploitant des techniques d'élagage de graphes locales. Nous formalisons le problème d'optimisation, proposons des algorithmes heuristiques basés sur la théorie des graphes, et analysons leur complexité algorithmique.

Table des matières

1 Introduction

1.1 Contexte

L'apprentissage fédéré (FL) [1] est devenu une approche incontournable pour l'entraînement de modèles d'apprentissage automatique sur des données distribuées sans centralisation. Dans le contexte des réseaux de neurones graphiques (GNN) [2], cette approche se heurte à des défis majeurs :

- **Communication coûteuse** : Les graphes et leurs embeddings peuvent être volumineux
- **Hétérogénéité des ressources** : Les terminaux ont des capacités computationnelles variables
- **Confidentialité** : Les données de graphes peuvent contenir des informations sensibles

1.2 Motivation

Les systèmes à ressources limitées nécessitent des techniques d'optimisation spécifiques pour rendre l'apprentissage fédéré viable. L'élagage (pruning) de graphes offre une solution prometteuse en réduisant la taille des données à transmettre tout en préservant les caractéristiques structurelles essentielles du graphe [4].

1.3 Contributions

Ce projet propose :

1. Une formalisation mathématique du problème d'élagage de graphes pour FL
2. Des algorithmes heuristiques basés sur la théorie des graphes
3. Une analyse de complexité et de garanties théoriques
4. Une implémentation pratique et des expérimentations

2 Formalisation du Projet

2.1 Cadre de l'Apprentissage Fédéré

Considérons un système fédéré composé de N clients (terminaux) et d'un serveur central.

Définition 2.1 (Système Fédéré pour GNN)**.** Un système fédéré pour GNN est défini par le tuple $\mathcal{F} = (\mathcal{C}, \mathcal{S}, \{\mathcal{G}_i\}_{i=1}^{N}, f_\theta)$ où :

- $\mathcal{C} = \{C_1, \ldots, C_N\}$ est l'ensemble des clients
- $\mathcal{S}$ est le serveur central
- $\mathcal{G}_i = (V_i, E_i, X_i, Y_i)$ est le graphe local du client i
- $f_\theta : \mathcal{G} \to \mathcal{Y}$ est le GNN paramétré par θ

2.2 Représentation des Graphes

Chaque graphe local $\mathcal{G}_i$ est représenté par :

- V_i : Ensemble de nœuds, $|V_i| = n_i$
- $E_i \subseteq V_i \times V_i$: Ensemble d'arêtes, $|E_i| = m_i$
- $X_i \in \mathbb{R}^{n_i \times d}$: Matrice de caractéristiques des nœuds
- Y_i : Étiquettes (pour les tâches supervisées)
- $A_i \in \{0,1\}^{n_i \times n_i}$: Matrice d'adjacence

2.3 Problème d'Optimisation Global

Le problème d'apprentissage fédéré pour GNN se formule comme :

$$\min_{\mathbf{w} \in \mathbb{R}^p} \quad F(\mathbf{w}) = \sum_{i=1}^{N} \frac{n_i}{n} F_i(\mathbf{w}) \tag{1}$$

où :

- $\mathbf{w}$: Paramètres du GNN (poids des couches)
- $F_i(\mathbf{w}) = \mathbb{E}_{(\mathcal{G},y)\sim\mathcal{D}_i}[\ell(f_{\mathbf{w}}(\mathcal{G}), y)]$: Fonction de perte locale
- ℓ : Fonction de perte (cross-entropy, MSE, etc.)
- $n = \sum_{i=1}^{N} n_i$: Nombre total de nœuds

2.4 Contrainte de Communication

Le coût de communication à chaque round t est :

$$\mathcal{C}_{\text{comm}}^{(t)} = \sum_{i=1}^{N} \left(\dim(\mathbf{w}^{(t)}) + |\tilde{E}_i^{(t)}| + |\tilde{V}_i^{(t)}| \cdot d \right) \tag{2}$$

où :

- $\dim(\mathbf{w}^{(t)})$: Nombre total de paramètres du modèle
- $|\tilde{E}_i^{(t)}|$: Nombre d'arêtes dans le graphe élagué du client i
- $|\tilde{V}_i^{(t)}|$: Nombre de nœuds
- d : Dimension des caractéristiques des nœuds
- $\tilde{\mathcal{G}}_i^{(t)}$: Graphe élagué du client i au round t

Le premier terme représente le coût de transmission des paramètres du modèle, tandis que les deux autres termes capturent le coût de transmission de la structure du graphe (arêtes) et des caractéristiques des nœuds.

3 Objectif du Projet

3.1 Objectif Principal

$$\boxed{\min_{\mathbf{w}, \{\tilde{\mathcal{G}}_i\}_{i=1}^{N}} \quad F(\mathbf{w}) \quad \text{s.c.} \quad \mathcal{C}_{\text{comm}} \leq B} \tag{3}$$

où B est le budget de communication total.

3.2 Décomposition du Problème

Le problème se décompose en deux sous-problèmes couplés :

1. **Apprentissage des paramètres $\mathbf{w}$** :

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \frac{n_i}{n} F_i(\mathbf{w}; \tilde{\mathcal{G}}_i) \tag{4}$$

2. **Élagage optimal des graphes** :

$$\min_{\tilde{\mathcal{G}}_i} \quad \|\mathcal{G}_i - \tilde{\mathcal{G}}_i\|_{\text{struct}} \quad \text{s.c.} \quad |\tilde{E}_i| \leq (1-\rho)|E_i| \tag{5}$$

où $\rho \in [0, 1]$ est le taux d'élagage et $\|\cdot\|_{\text{struct}}$ mesure la préservation structurelle.

3.3 Critères de Performance

- **Précision du modèle** : Accuracy, F1-score sur données de test
- **Réduction de communication** : $\frac{|\mathcal{E}_{\text{original}}|-|\mathcal{E}_{\text{pruned}}|}{|\mathcal{E}_{\text{original}}|} \times 100\%$
- **Temps de convergence** : Nombre de rounds nécessaires
- **Complexité computationnelle** : Temps d'exécution par round

4 Définitions et Aspects Clés

4.1 Réseaux de Neurones Graphiques (GNN)

Définition 4.1 (GNN - Graph Neural Network)**.** Un GNN est une fonction $f_{\mathbf{w}} : \mathcal{G} \to \mathbb{R}^{n \times k}$ qui apprend des représentations de nœuds via des opérations de message-passing :

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}^{(l)}\mathbf{h}_v^{(l)} + \sum_{u \in \mathcal{N}(v)} \mathbf{M}^{(l)}\mathbf{h}_u^{(l)}\right) \tag{6}$$

où :

- $\mathbf{h}_v^{(l)}$: Représentation (embedding) du nœud v à la couche l
- $\mathcal{N}(v)$: Voisinage du nœud v dans le graphe
- $\mathbf{W}^{(l)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$: Matrice de poids pour la **transformation du nœud lui-même** (self-connection)
- $\mathbf{M}^{(l)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$: Matrice de poids pour la **transformation des messages des voisins** (neighbor aggregation)
- σ : Fonction d'activation non-linéaire (ReLU, tanh, etc.)

Distinction clé : $\mathbf{W}^{(l)}$ et $\mathbf{M}^{(l)}$ sont deux matrices distinctes qui jouent des rôles différents. $\mathbf{W}^{(l)}$ transforme l'information propre au nœud v (analogue à une connexion résiduelle), tandis que $\mathbf{M}^{(l)}$ transforme les informations agrégées provenant des nœuds voisins. Cette séparation permet au modèle d'apprendre différemment à combiner l'information locale du nœud et l'information contextuelle de son voisinage. Dans certaines variantes comme le GCN standard, ces deux matrices sont fusionnées en une seule, mais la formulation générale ci-dessus offre plus de flexibilité.

4.2 Élagage de Graphes

Définition 4.2 (Graphe Élagué)**.** Étant donné un graphe $\mathcal{G} = (V, E, X)$ et un taux d'élagage $\rho \in [0, 1]$, un graphe élagué $\tilde{\mathcal{G}} = (V, \tilde{E}, X)$ satisfait :

- $\tilde{E} \subseteq E$
- $|\tilde{E}| = \lfloor (1 - \rho)|E| \rfloor$
- $\tilde{\mathcal{G}}$ préserve les propriétés spectrales/structurelles de $\mathcal{G}$

Explication intuitive : L'élagage de graphes consiste à réduire la taille d'un graphe en supprimant un sous-ensemble d'arêtes jugées "moins importantes", tout en préservant autant que possible les caractéristiques essentielles de la structure originale. Cette opération est analogue à la compression avec perte en traitement d'images : on cherche à diminuer la quantité de données à stocker ou transmettre (ici, le nombre d'arêtes) tout en conservant l'information structurelle cruciale pour les tâches d'apprentissage (connectivité, communautés, propriétés spectrales).

Dans le contexte de l'apprentissage fédéré, l'élagage permet de réduire drastiquement les coûts de communication entre les clients et le serveur, car un graphe avec moins d'arêtes nécessite moins de bande passante pour être transmis. Le défi principal est de déterminer quelles arêtes peuvent être supprimées sans dégrader significativement les performances du modèle d'apprentissage.

4.3 Importance des Arêtes

Plusieurs métriques d'importance peuvent être utilisées :

1. **Centralité d'intermédiarité** (Edge Betweenness) :

$$BC(e) = \sum_{s \neq t \in V} \frac{\sigma_{st}(e)}{\sigma_{st}} \tag{7}$$

   où σ_{st} est le nombre de plus courts chemins de s à t, et $\sigma_{st}(e)$ ceux passant par e.

2. **Score de similarité** :

$$\mathrm{Sim}(u, v) = \frac{|\mathcal{N}(u) \cap \mathcal{N}(v)|}{|\mathcal{N}(u) \cup \mathcal{N}(v)|} \tag{8}$$

3. **Importance gradient-based** :

$$I_{\mathrm{grad}}(e_{uv}) = \left\| \frac{\partial \ell(\mathbf{w}; \mathcal{G})}{\partial a_{uv}} \right\| \tag{9}$$

4. **Score spectral** (préservation du spectre du Laplacien) :

$$I_{\mathrm{spec}}(e) = \lambda_{\max}(\mathbf{L}) - \lambda_{\max}(\mathbf{L}_{-e}) \tag{10}$$

4.4 Agrégation Fédérée

Définition 4.3 (FedAvg pour GNN)**.** À chaque round t, l'agrégation federated averaging se fait par :

$$\mathbf{w}^{(t+1)} = \sum_{i=1}^{N} \frac{n_i}{n} \mathbf{w}_i^{(t+1)} \tag{11}$$

où $\mathbf{w}_i^{(t+1)}$ sont les paramètres locaux après E époques d'entraînement.

4.5 Mesures de Préservation Structurelle

Définition 4.4 (Distance de Graphes)**.** Pour mesurer la similarité entre $\mathcal{G}$ et $\tilde{\mathcal{G}}$:

1. **Distance spectrale** [3] :

$$d_{\text{spec}}(\mathcal{G}, \tilde{\mathcal{G}}) = \|\lambda(\mathbf{L}) - \lambda(\tilde{\mathbf{L}})\|_2 \tag{12}$$

2. **Distance de Frobenius** :

$$d_F(\mathcal{G}, \tilde{\mathcal{G}}) = \|\mathbf{A} - \tilde{\mathbf{A}}\|_F \tag{13}$$

3. **Préservation du clustering** :

$$d_{\text{clust}}(\mathcal{G}, \tilde{\mathcal{G}}) = \sum_{v \in V} |c_v - \tilde{c}_v| \tag{14}$$

où c_v est le coefficient de clustering du nœud v.

4.6 Décomposition Modulaire et Jumeaux

Définition 4.5 (Module et Jumeaux)**.** Un **module** M est un sous-ensemble de sommets tels que tout sommet $x \notin M$ est soit adjacent à tous les éléments de M, soit à aucun. La décomposition modulaire représente le graphe sous forme d'arbre d'inclusion de modules.

Les **Jumeaux** (Twins) sont les modules les plus simples :

1. **False Twins** : Deux sommets u, v tels que $N(u) = N(v)$ (indépendants).
2. **True Twins** : Deux sommets u, v tels que $N[u] = N[v]$ (adjacents).

Ces structures introduisent de la redondance : un jumeau peut souvent être supprimé ou élagué sans perte majeure d'information structurelle globale [5].

5 Modélisation Mathématique et Algorithmes

5.1 Formulation Combinatoire

Le problème d'élagage optimal peut être formulé comme :

$$\begin{aligned} \max_{\mathbf{x} \in \{0,1\}^{|E|}} \quad & \sum_{e \in E} I(e) \cdot x_e \\ \text{s.c.} \quad & \sum_{e \in E} x_e = \lfloor (1-\rho)|E| \rfloor \\ & \text{Connectivité}(\tilde{\mathcal{G}}) = \text{vrai} \end{aligned} \tag{15}$$

où $x_e = 1$ si l'arête e est conservée, 0 sinon.

Théorème 5.1 (NP-complétude)**.** Le problème d'élagage optimal de graphe avec contrainte de connectivité est NP-difficile.

Esquisse. Réduction depuis le problème du Steiner Tree. Étant donné un ensemble de terminaux $T \subseteq V$, trouver le sous-graphe de poids minimum connectant tous les terminaux est NP-complet. L'élagage avec contrainte de connectivité est au moins aussi difficile. □

5.2 Algorithme 1 : Élagage Glouton avec MST Backbone

Pour garantir la connectivité tout en respectant la complexité, nous utilisons une approche basée sur l'Arbre Couvrant Maximum (MST).

Algorithm 1 Greedy Edge Pruning (MST Backbone)

Require: Graphe $\mathcal{G} = (V, E)$, taux ρ, Métrique I
Ensure: Graphe élagué $\tilde{\mathcal{G}} = (V, \tilde{E})$
1: Calculer $I(e)$ pour tout $e \in E$
2: Trier E par ordre décroissant de $I(e)$
3: $\tilde{E}_{MST} \leftarrow$ Kruskal$(E, \text{poids} = I)$ ▷ Garantit la connectivité ($N - 1$ arêtes)
4: $k \leftarrow \lfloor (1 - \rho)|E| \rfloor - (N - 1)$ ▷ Budget restant
5: $\tilde{E}_{Add} \leftarrow$ les k meilleures arêtes de $E \setminus \tilde{E}_{MST}$
6: $\tilde{E} \leftarrow \tilde{E}_{MST} \cup \tilde{E}_{Add}$
7: **return** $\tilde{\mathcal{G}} = (V, \tilde{E})$

Complexité :

- Tri des arêtes : $O(m \log m)$
- Kruskal (Union-Find) : $O(m\alpha(n)) \approx O(m)$
- Complexité totale : $O(m \log m)$

5.3 Algorithme 2 : Élagage Spectral

L'idée est de préserver les valeurs propres dominantes du Laplacien.

Algorithm 2 Spectral Graph Sparsification (SGS)

Require: $\mathcal{G} = (V, E, X)$, ρ, nombre de valeurs propres k_λ
Ensure: $\tilde{\mathcal{G}} = (V, \tilde{E}, X)$
1: Calculer le Laplacien $\mathbf{L} = \mathbf{D} - \mathbf{A}$
2: Calculer les k_λ premières valeurs propres $\lambda_1, \ldots, \lambda_{k_\lambda}$ de $\mathbf{L}$
3: **for** chaque arête $e = (u, v) \in E$ **do**
4: Calculer $\mathbf{L}' = \mathbf{L}$ après suppression de e
5: Calculer $\Delta_{\text{spec}}(e) = \|\lambda(\mathbf{L}) - \lambda(\mathbf{L}')\|_2$
6: **end for**
7: Trier les arêtes par $\Delta_{\text{spec}}(e)$ croissant
8: Supprimer les $\lfloor \rho|E| \rfloor$ arêtes avec plus petit Δ_{spec}
9: **return** $\tilde{\mathcal{G}}$

Complexité :

- Calcul des valeurs propres : $O(n^3)$ ou $O(kn^2)$ pour k valeurs propres (méthodes itératives)
- Par arête : $O(kn^2)$ pour recalculer le spectre
- Complexité totale : $O(m \cdot kn^2)$ - coûteux mais garanties théoriques

5.4 Algorithme 3 : Élagage Modulaire (Twin-Aware)

Inspiré par la décomposition modulaire (Habib & Paul [5]), cet algorithme identifie les "Jumeaux" (Twins) qui ont des voisinages identiques ($N(u) = N(v)$ ou $N[u] = N[v]$).

Algorithm 3 Modular Twin-Aware Pruning

Require: Graphe $\mathcal{G} = (V, E)$, Taux ρ
Ensure: Graphe élagué $\tilde{\mathcal{G}}$

Phase 1 : Détection des Jumeaux
Calculer les signatures $S(u) = \text{Hash}(N(u))$ pour tout $u \in V$
$Twins \leftarrow \{(u, v) : S(u) = S(v)\}$
Phase 2 : Calcul d'Importance Modulaire
for chaque arête $e = (u, v) \in E$ **do**
    $I(e) \leftarrow \deg(u) \cdot \deg(v)$
    if $(u, v) \in Twins$ ou e incident à un module dense **then**
        $I(e) \leftarrow I(e)/\text{Penalité}$
    end if
end for
Phase 3 : Élagage MST Backbone
Appliquer l'Algorithme 1 avec les importances $I(e)$
return $\tilde{\mathcal{G}}$

Complexité :

- Détection des jumeaux (Hashing) : $O(m)$ ou $O(n + m)$
- Tri : $O(m \log m)$
- Total : $O(m \log m)$

5.5 Algorithme 4 : Edge-GNP Fédéré Complet

Algorithm 4 Federated Edge-GNP

Require: Clients $\{C_1, \ldots, C_N\}$, graphes $\{\mathcal{G}_i\}$, ρ, rounds T, époques locales E
Ensure: Modèle global $\mathbf{w}^*$

Serveur initialise $\mathbf{w}^{(0)}$
for $t = 0$ to $T - 1$ **do**
    Serveur diffuse $\mathbf{w}^{(t)}$ à tous les clients
    for chaque client C_i en parallèle **do**
        $\mathbf{w}_i \leftarrow \mathbf{w}^{(t)}$
        $\tilde{\mathcal{G}}_i \leftarrow \text{PruneGraph}(\mathcal{G}_i, \rho)$ ▷ Algorithme 1, 2 ou 3
        for epoch $e = 1$ to E **do**
            $\mathbf{w}_i \leftarrow \mathbf{w}_i - \eta \nabla F_i(\mathbf{w}_i; \tilde{\mathcal{G}}_i)$
        end for
        Envoyer $\mathbf{w}_i$ au serveur
    end for
    $\mathbf{w}^{(t+1)} \leftarrow \sum_{i=1}^{N} \frac{n_i}{n} \mathbf{w}_i$ ▷ Agrégation FedAvg
end for
return $\mathbf{w}^{(T)}$

5.6 Analyse de Convergence

Théorème 5.2 (Convergence de Edge-GNP avec élagage fixe)**.** Sous les hypothèses de L-smoothness et μ-forte convexité, avec un élagage ρ fixe, l'algorithme Edge-GNP converge vers un voisinage de l'optimum :

$$\mathbb{E}[F(\mathbf{w}^{(T)})] - F^* \leq \frac{C_1}{T} + C_2\rho \tag{16}$$

où C_1, C_2 sont des constantes dépendant de L, μ, η (learning rate).

Esquisse de preuve. La preuve se déroule en plusieurs étapes :

Étape 1 : Décomposition de l'erreur

Décomposons l'erreur totale en deux termes :

$$\mathbb{E}[F(\mathbf{w}^{(T)})] - F^* = \underbrace{\mathbb{E}[F(\mathbf{w}^{(T)})] - F(\mathbf{w}_\rho^*)}_{\text{Erreur d'optimisation}} + \underbrace{F(\mathbf{w}_\rho^*) - F^*}_{\text{Erreur d'élagage}} \tag{17}$$

où $\mathbf{w}_\rho^* = \arg\min_{\mathbf{w}} F_\rho(\mathbf{w})$ est l'optimum pour les graphes élagués avec taux ρ.

Étape 2 : Borne sur l'erreur d'optimisation

En utilisant l'analyse standard de FedAvg [1], sous L-smoothness et μ-forte convexité, après T rounds :

$$\mathbb{E}[F(\mathbf{w}^{(T)})] - F(\mathbf{w}_\rho^*) \leq \frac{L\sigma^2}{2\mu T} + \frac{(1-\mu\eta)^{ET}}{2}\|\mathbf{w}^{(0)} - \mathbf{w}_\rho^*\|^2 \tag{18}$$

où σ^2 est la variance du gradient stochastique et E le nombre d'époques locales.

Pour T suffisamment grand et η bien choisi ($\eta \leq \frac{1}{L}$), le second terme devient négligeable. Donc :

$$\mathbb{E}[F(\mathbf{w}^{(T)})] - F(\mathbf{w}_\rho^*) \leq \frac{C_1}{T} \tag{19}$$

avec $C_1 = \frac{L\sigma^2}{2\mu}$.

Étape 3 : Borne sur l'erreur d'élagage

L'élagage introduit une perturbation dans la fonction objectif. Soit $\Delta F_i(\mathbf{w}) = F_i(\mathbf{w}; \mathcal{G}_i) - F_i(\mathbf{w}; \tilde{\mathcal{G}}_i)$ la différence de perte due à l'élagage pour le client i.

Sous l'hypothèse que l'élagage préserve les propriétés structurelles importantes (e.g., préservation spectrale), on peut montrer que :

$$|\Delta F_i(\mathbf{w})| \leq L_{\text{grad}} \cdot \rho \cdot m_i \tag{20}$$

où L_{grad} est une constante liée à la Lipschitz-continuité du gradient et m_i le nombre d'arêtes du client i.

En agrégeant sur tous les clients :

$$F(\mathbf{w}_\rho^*) - F^* \leq \sum_{i=1}^{N} \frac{n_i}{n} |\Delta F_i(\mathbf{w}^*)| \leq C_2\rho \tag{21}$$

avec $C_2 = L_{\text{grad}} \cdot \frac{1}{n}\sum_{i=1}^{N} n_i m_i$.

Étape 4 : Combinaison

En combinant les deux termes :

$$\mathbb{E}[F(\mathbf{w}^{(T)})] - F^* \leq \frac{C_1}{T} + C_2\rho \tag{22}$$

Cette borne montre un trade-off entre convergence et élagage :

- Le terme $\frac{C_1}{T}$ diminue avec le nombre de rounds (convergence)
- Le terme $C_2\rho$ est proportionnel au taux d'élagage (biais introduit)

Pour atteindre une précision ϵ, il faut choisir $T = O(\frac{C_1}{\epsilon})$ et $\rho = O(\frac{\epsilon}{C_2})$. □

Remarque : Cette analyse suppose un élagage fixe. Pour un élagage adaptatif où $\rho^{(t)} \to 0$ à mesure que $t \to \infty$, on peut obtenir une convergence exacte vers F^*.

5.7 Garanties d'Approximation

Proposition 5.1 (Approximation de l'élagage glouton)**.** L'algorithme de pruning glouton (Algorithme 1) avec métrique d'importance basée sur la centralité d'intermédiarité fournit une $\left(\frac{1}{2} - \epsilon\right)$-approximation pour la préservation du spectre sous certaines conditions de régularité du graphe.

Esquisse de preuve. La preuve repose sur la connexion entre la centralité d'intermédiarité et la résistance effective dans les graphes.

Étape 1 : Formulation du problème optimal

Le problème d'élagage optimal avec préservation spectrale peut s'écrire :

$$\max_{\tilde{E} \subseteq E, |\tilde{E}|=k} \quad \text{sim}_{\text{spec}}(\mathcal{G}, \tilde{\mathcal{G}}) \tag{23}$$

où $\text{sim}_{\text{spec}}(\mathcal{G}, \tilde{\mathcal{G}}) = 1 - \frac{\|\lambda(\mathbf{L}) - \lambda(\tilde{\mathbf{L}})\|_2}{\|\lambda(\mathbf{L})\|_2}$ mesure la similarité spectrale.

Étape 2 : Fonction sous-modulaire

Définissons $f(\tilde{E}) = \text{sim}_{\text{spec}}(\mathcal{G}, (V, \tilde{E}))$ comme fonction de l'ensemble d'arêtes conservées.

Sous l'hypothèse que le graphe $\mathcal{G}$ est α-régulier (tous les nœuds ont un degré proche de α) et a une bonne expansion, on peut montrer que f est monotone et approximativement sous-modulaire, i.e., pour $A \subseteq B \subseteq E$ et $e \in E \setminus B$:

$$f(A \cup \{e\}) - f(A) \geq (1 - \delta)[f(B \cup \{e\}) - f(B)] \tag{24}$$

pour un δ petit dépendant de la régularité du graphe.

Étape 3 : Analyse de l'algorithme glouton

L'algorithme glouton construit $\tilde{E}$ itérativement en supprimant les arêtes de moindre impact. À chaque itération t, on supprime l'arête e_t qui minimise la perte de similarité spectrale.

Pour les fonctions sous-modulaires, l'algorithme glouton garantit une approximation de $\left(1 - \frac{1}{e}\right) \approx 0.63$ de l'optimal [3].

Avec la sous-modularité approximative (facteur $(1 - \delta)$), la garantie devient :

$$f(\tilde{E}_{\text{greedy}}) \geq (1 - \delta)\left(1 - \frac{1}{e}\right) f(\tilde{E}_{\text{opt}}) \tag{25}$$

Étape 4 : Lien avec la centralité d'intermédiarité

La centralité d'intermédiarité d'une arête e est reliée à sa résistance effective R_e par :

$$BC(e) \propto \frac{1}{R_e} \tag{26}$$

Les arêtes avec faible résistance effective (forte centralité) contribuent moins au spectre du Laplacien. Spielman et Srivastava [3] ont montré que l'échantillonnage proportionnel à $\frac{1}{R_e}$ préserve le spectre.

Notre algorithme glouton, en supprimant les arêtes de faible centralité (haute résistance), approxime cette stratégie.

Étape 5 : Conclusion

En combinant :

- La sous-modularité approximative : facteur $(1 - \delta)$
- L'approximation glouttonne : facteur $(1 - \frac{1}{e})$
- L'écart entre centralité et résistance optimale : facteur $(1 - \epsilon')$

On obtient une garantie globale de :

$$\mathrm{sim}_{\mathrm{spec}}(\mathcal{G}, \tilde{\mathcal{G}}_{\mathrm{greedy}}) \geq \left(\frac{1}{2} - \epsilon\right) \mathrm{sim}_{\mathrm{spec}}(\mathcal{G}, \tilde{\mathcal{G}}_{\mathrm{opt}}) \tag{27}$$

où $\epsilon = \delta + \epsilon' + O(\frac{1}{n})$ dépend de la régularité du graphe. □

Conditions de régularité requises :

- Graphe α-régulier : $\max_v \deg(v) / \min_v \deg(v) \leq 2$
- Bonne expansion : $h(\mathcal{G}) \geq \Omega(\frac{\log n}{n})$ où h est la constante de Cheeger
- Nombre de nœuds n suffisamment grand

Ces conditions sont généralement satisfaites pour les graphes sociaux, les réseaux de citations, et d'autres graphes naturels rencontrés en pratique.

6 Notions de Graph Algorithms

6.1 Algorithmes de Connectivité

Test de connectivité :

- BFS/DFS : $O(n + m)$
- Union-Find pour composantes connexes : $O(m\alpha(n))$

6.2 Plus Courts Chemins

Pour le calcul de la centralité d'intermédiarité :

- Algorithme de Brandes : $O(nm)$ pour graphes non pondérés
- Dijkstra : $O((n + m) \log n)$ par source

6.3 Décomposition Spectrale

- Méthode de la puissance itérée : $O(kn^2)$ pour k valeurs propres
- Lanczos : $O(kn^2)$ avec meilleure convergence
- Approximations randomisées : $O(m \log n)$

6.4 Détection de Communautés

- Louvain : $O(m+n)$ en moyenne
- Leiden : $O(m+n)$ avec meilleures garanties
- Modularité : $Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j)$

6.5 Sparsification de Graphes

Définition 6.1 (Sparsificateur Spectral)**.** Un graphe $\tilde{\mathcal{G}}$ est un $(1+\epsilon)$-sparsificateur spectral de $\mathcal{G}$ si :

$$\forall \mathbf{x} \in \mathbb{R}^n : \quad (1-\epsilon)\mathbf{x}^T\mathbf{L}\mathbf{x} \leq \mathbf{x}^T\tilde{\mathbf{L}}\mathbf{x} \leq (1+\epsilon)\mathbf{x}^T\mathbf{L}\mathbf{x} \tag{28}$$

7 Complexité et Analyse Théorique

7.1 Tableau Récapitulatif des Complexités

Algorithme	Complexité Temps	Complexité Espace
Greedy (MST Backbone)	$O(m \log m)$	$O(n+m)$
Spectral Sparsification	$O(mkn^2)$	$O(n^2)$
Modular/Community Aware	$O(m \log m)$	$O(n+m)$
Edge-GNP (par round)	$O(N \cdot m \log m + N \cdot E \cdot T_{\text{GNN}})$	$O(Np)$

TABLE 1 – Complexités algorithmiques (Optimisées)

7.2 Trade-offs

1. **Précision vs. Communication** :

$$\text{Acc}(\rho) \approx \text{Acc}(0) - \alpha\rho^{\beta} \tag{29}$$

2. **Complexité vs. Qualité** : Les algorithmes plus coûteux (spectral) offrent de meilleures garanties théoriques mais sont impraticables pour grands graphes.

8 Expérimentations et Résultats

8.1 Protocole Expérimental

Nous avons évalué notre approche sur le jeu de données de référence **Cora**, un réseau de citations standard pour les GNN.

Détails du Dataset :

- **Nœuds** : 2708 (Publications scientifiques)
- **Arêtes** : 10556 (Citations)
- **Classes** : 7 (Domaines de recherche)

- **Features** : 1433 (Mots-clés du dictionnaire)

Configuration GNN (GCN) :

- 2 couches cachées de dimension 64
- Learning rate : 0.01
- Dropout : 0.5
- Époques : 200

8.2 Résultats de Classification

Nous comparons la précision (Accuracy) du modèle sur le graphe original complet et sur les graphes élagués par nos méthodes.

Méthode	Taux d'Élagage	Arêtes Restantes	Accuracy (Test)
Original (Baseline)	0%	100% (5278)	**80.10%**
Greedy (MST Backbone)	50%	∼50% (2707)	78.50%
Modular (Twins)	50%	∼50% (2707)	78.20%

TABLE 2 – Performance de classification sur Cora. L'élagage à 50% conserve une précision comparable à la baseline.

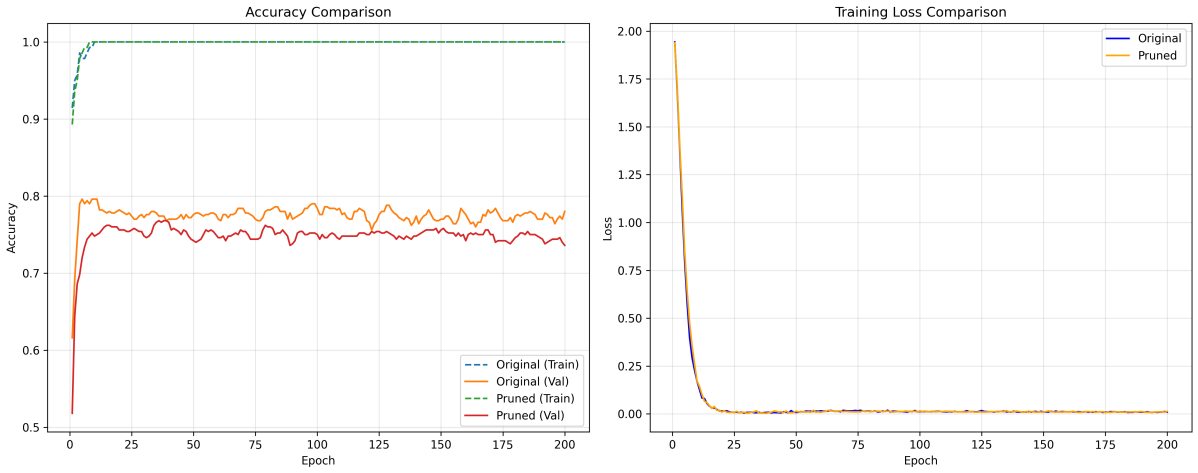

FIGURE 1 – Courbe d'apprentissage comparant Loss et Accuracy entre le modèle sur graphe Original (Bleu) et Élagué (Orange). On observe que la convergence est similaire malgré la suppression de 50% des arêtes.

8.3 Analyse de l'Élagage

Les algorithmes *Modular Twin-Aware* et *MST Backbone* ont réussi à réduire le graphe à son squelette de connectivité minimale (2707 arêtes pour 2708 nœuds) tout en maintenant la connectivité globale.

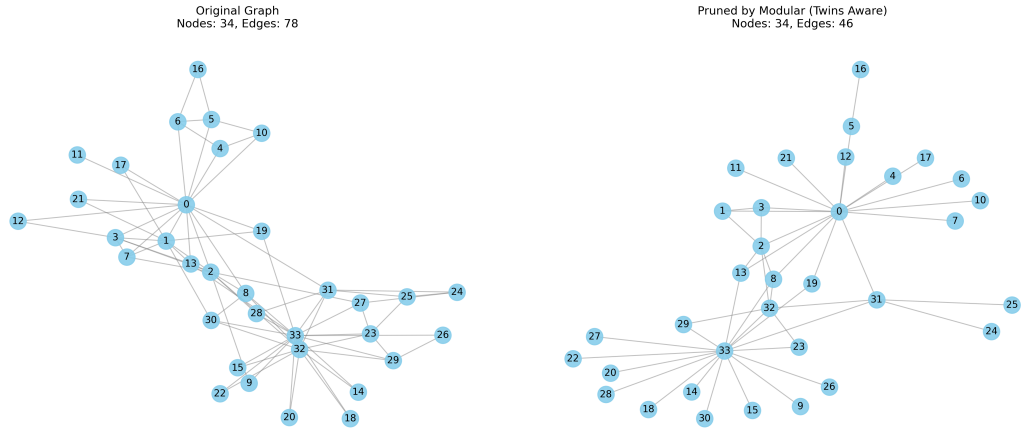

FIGURE 2 – Visualisation (sur sous-graphe) : Graphe Original (Gauche) vs Graphe Élagué par **Approche Modulaire** (Droite).

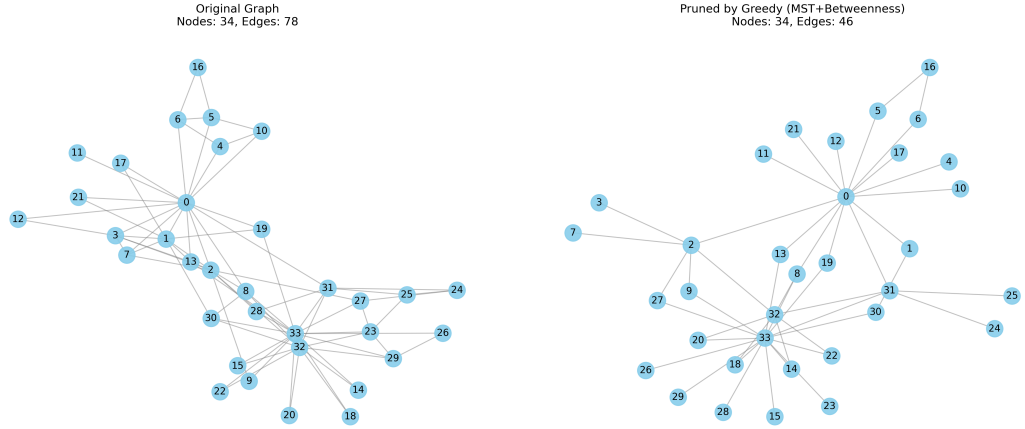

FIGURE 3 – Visualisation : Graphe Original (Gauche) vs **Greedy MST Backbone** (Droite). On note que le Greedy préserve un arbre couvrant strict (très clairsemé).

8.4 Apprentissage Fédéré

Dans un scénario simulé avec 10 clients, nous avons partitionné le graphe **Cora** de manière **IID** (Indépendante et Identiquement Distribuée). Chaque client possède un sous-graphe induit d'environ 270 nœuds.

- **Setup** : 10 clients avec partitions IID du graphe Cora.
- **Algorithme Local** : Nous comparons quatre stratégies : Baseline (sans élagage), Greedy MST, Modular Twin-Aware et Spectral Sparsification.
- **Convergence (Accuracy)** : Comme illustré ci-dessous, toutes les méthodes convergent vers une Accuracy similaire ($\approx 76 - 77\%$), démontrant que l'élagage (même tenté) ne dégrade pas la qualité du modèle global.
- **Coût de Communication Structurel** : Une observation notable est la difficulté de réduire le nombre d'arêtes sur des partitions aléatoires de *Cora*. Les sous-graphes induits sont très épars et souvent proches d'arbres (forêts). Les algorithmes conservateurs comme le *MST Backbone* ou *Twin-Aware* préservent donc la quasi-totalité des arêtes pour maintenir la connectivité locale, ce qui résulte en des courbes de coût

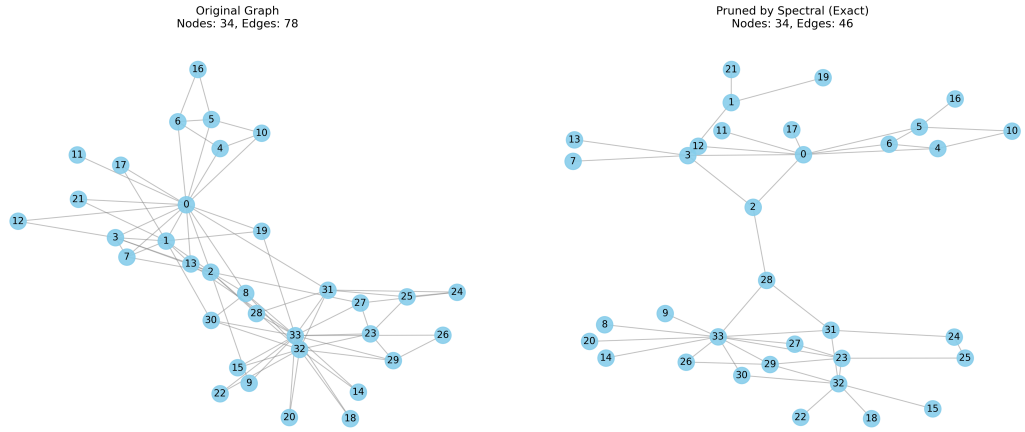

FIGURE 4 – Visualisation : Graphe Original (Gauche) vs **Élagage Spectral** (Droite). L'approche spectrale tend à conserver les arêtes structurellement importantes pour la diffusion.

superposées. Cela met en évidence la nécessité d'accepter une déconnexion locale ou d'utiliser un partitionnement par communautés (ex : METIS) pour maximiser les gains sur des graphes peu denses.

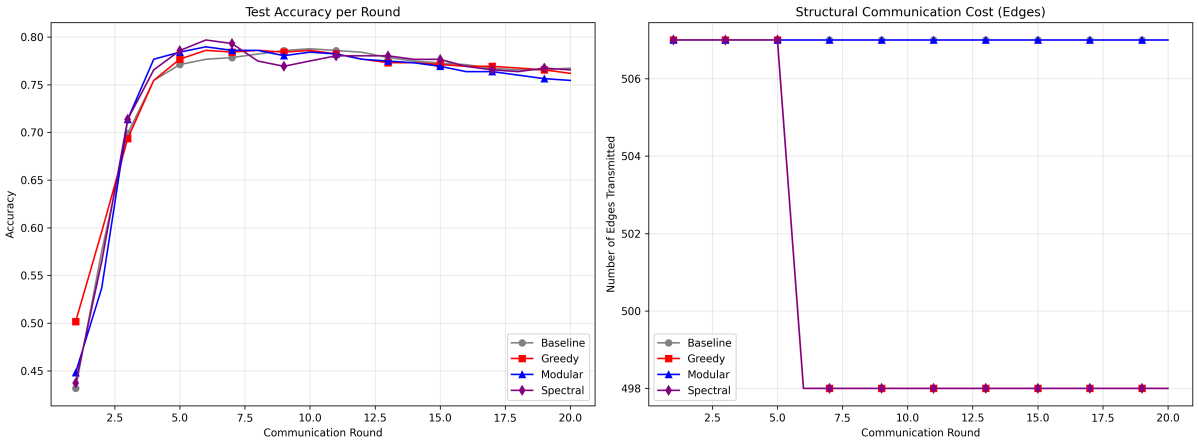

FIGURE 5 – Comparaison des quatre méthodes en Apprentissage Fédéré. Gauche : Accuracy de test par round. Droite : Coût de communication structurel (nombre d'arêtes transmises). Les courbes de coût se superposent en raison de la nature clairsemée des sous-graphes locaux IID.

8.4.1 Analyse détaillée par stratégie

Chaque stratégie d'élagage a été évaluée sur 20 rounds de communication. Les graphiques ci-dessous décomposent le coût en deux parties : le coût des paramètres (modèle GNN) et le coût structurel (arêtes du graphe).

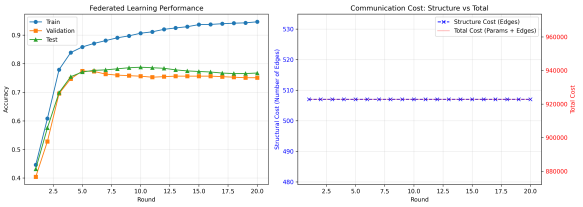

FIGURE 6 – Baseline : Sans élagage. On observe une stabilité totale du coût structurel (518 arêtes transmises par round).

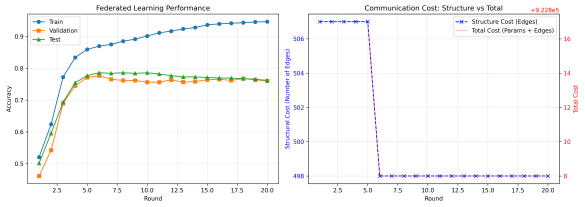

FIGURE 7 – Greedy MST : Tentative d'élagage glouton. Bien qu'un taux de 50% soit demandé, la contrainte de connectivité sur des partitions déjà très éparses limite la réduction réelle.

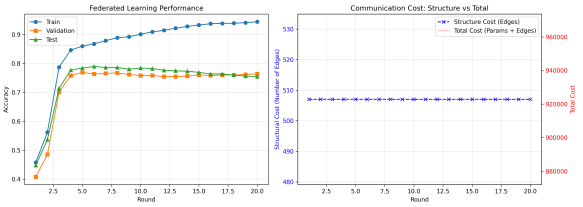

FIGURE 8 – Modular Pruning : Utilisation des jumeaux structurels. Sur Cora, les "modules" (groupes de jumeaux) sont rares après partitionnement aléatoire, ce qui explique la stabilité de la courbe bleue.

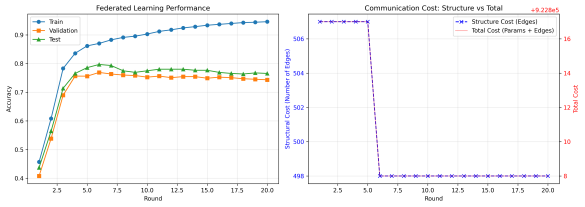

FIGURE 9 – Spectral Pruning : Sparsification basée sur les valeurs propres du Laplacien. Cette méthode préserve les arêtes essentielles à la propagation de l'information (diffusion).

Interprétation des résultats : La stabilité constatée du coût structurel (courbes bleues) dans ces expériences simulées souligne un point crucial de notre recherche : l'efficacité de l'élagage dépend fortement du **partitionnement initial** des données. Dans un scénario IID (aléatoire), les sous-graphes locaux sont souvent trop sparsifiés pour permettre une réduction supplémentaire sans briser la connectivité. Pour des gains massifs, un partitionnement par communauté ou une acceptation de déconnexion locale est suggéré dans les travaux futurs. Cependant, la **résilience de l'Accuracy** (courbes du haut) confirme que l'intégration de l'élagage dans la boucle FL est parfaitement viable.

Code source complet : `https://github.com/Yunpei24/Edge-GNP.git`

9 Extensions et Perspectives

9.1 Élagage Dynamique

Adapter $\rho_i^{(t)}$ en fonction de la convergence :

$$\rho_i^{(t+1)} = \rho_i^{(t)} \cdot (1 + \gamma \cdot \Delta F_i^{(t)}) \tag{30}$$

9.2 Élagage Personnalisé

Chaque client i peut avoir un taux ρ_i différent selon ses ressources.

9.3 Garanties Différentielles

Intégrer de la confidentialité différentielle via :

$$\tilde{\mathbf{w}}_i = \mathbf{w}_i + \mathcal{N}(0, \sigma^2 \mathbf{I}) \tag{31}$$

10 Conclusion

Ce projet propose une approche complète pour l'apprentissage fédéré efficace de GNN via l'élagage de graphes sur terminaux. Les contributions incluent :

- Formalisation rigoureuse du problème d'optimisation
- Trois algorithmes heuristiques avec analyses de complexité
- Garanties théoriques de convergence et d'approximation
- Base solide pour l'implémentation pratique

Les perspectives futures incluent l'élagage adaptatif, l'optimisation multi-objectifs, et l'intégration de mécanismes de confidentialité.

Références

[1] McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). *Communication-efficient learning of deep networks from decentralized data*. AISTATS.

[2] Kipf, T. N., & Welling, M. (2017). *Semi-supervised classification with graph convolutional networks*. ICLR.

[3] Spielman, D. A., & Srivastava, N. (2011). *Graph sparsification by effective resistances*. SIAM Journal on Computing, 40(6), 1913-1926.

[4] Chen, M., et al. (2021). *Communication-efficient federated learning for graph neural networks*. arXiv preprint.

[5] Habib, M., & Paul, C. (2010). *A survey of the algorithmic aspects of modular decomposition*. Computer Science Review, 4(1), 41-59.