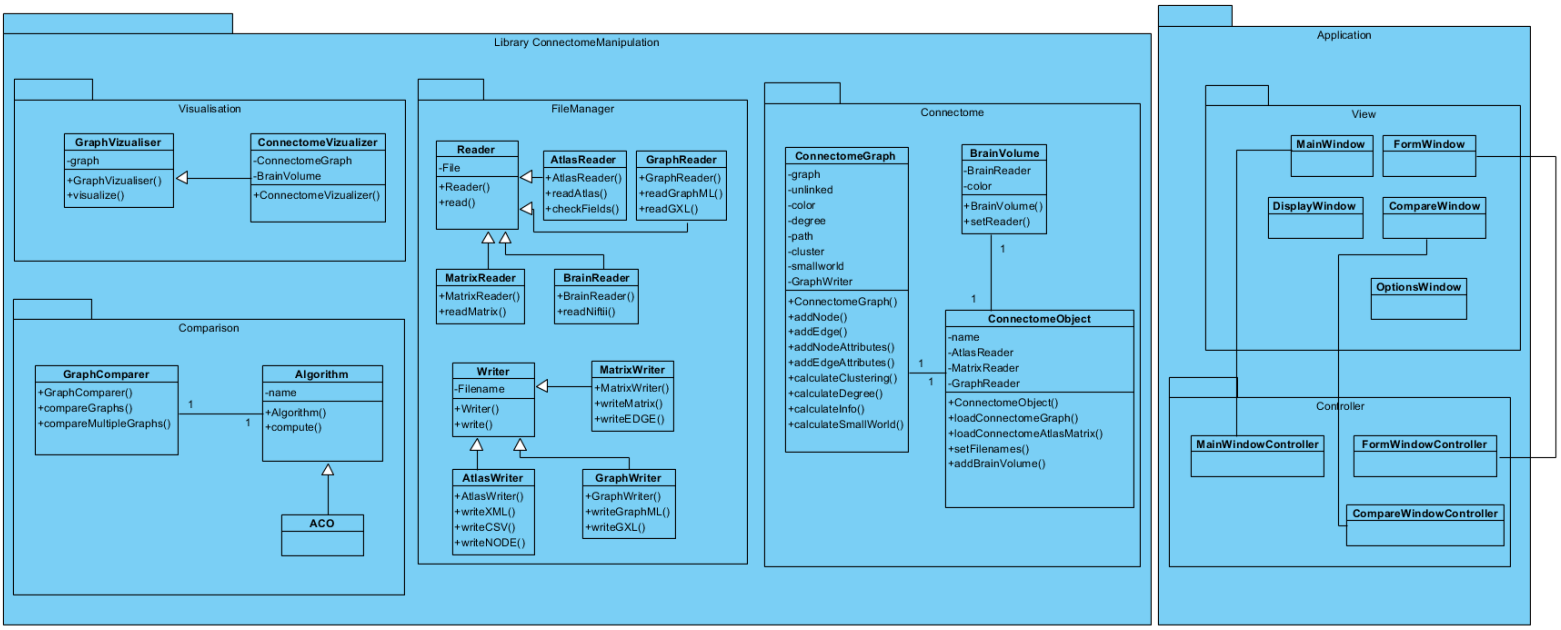
Cahier de développeur

I- Diagramme de classe et explications



Les classes du projet sont globalement réparties en deux différentes parties : les classes destinées à l'application, qui sont les classes d'interface graphique et de contrôleurs des interfaces, et les classes de manipulation de connectomes, ce qui englobe les classes de modélisation sous forme de graphe, de visualisation et de comparaison.

1) Module Connectome Manipulation

Le module de manipulation de connectomes peut être séparé en 4 modules distincts :

- Visualisation qui est le module en charge de visualiser les graphes
- Comparison qui est le module en charge de comparer des graphes
- FileManager qui est le module en charge de lecture et écriture de fichiers
- Connectome qui est le module de représentation de connectomes

Le module principal est le module Connectome qui va nous servir à représenter les objets connectomes qui sont utilisés par les autres modules. La classe principale est ConnectomeObject, un objet permettant de représenter un connectome et porte les informations et outils nécessaires à ce connectome. La classe ConnectomeGraph représente la représentation sous forme de graphe du ConnectomeObject.

Le module visualisation utilise la classe ConnectomeVizualiser qui prend en entrée un ConnectomeObject pour l'afficher.

Le module Comparison utilise le GraphComparer qui prend en entrée une liste de graphe à comparer (on utilise ici networkx pour les graphes) et applique un algorithme de comparaison pour les comparer. L'ACO (Ant Colony Optimization) est un algorithme de comparaison de graphes.

Le module FileManager gère les Reader et Writer des différents fichiers à manipuler, comme les matrices d'adjacence (MatrixReader/MatrixWriter), les atlas (AtlasReader/AtlasWriter), les fichiers graphes (GraphReader/GraphWriter) et les fichiers d'imagerie médicale (BrainReader).

Certaines classes ont été créées en prévision d'un travail futur, et les fonctionnalités ne sont pas encore développées. De même, certaines fonctions de classes autrement développées ne sont pas encore faite et ont été laissée à une continuation du projet. Les classes concernées sont :

- BrainReader
- BrainVolume
- GraphComparer
- Algorithm
- ACO
- MatrixWriter
- AtlasWriter

2) Module Application

Les classes de ce module concernent principalement des classes pour l'interface graphique. Dans le module View on trouve les classes de l'interface graphique faites avec Tkinter et dans le module Controller les classes qui permettent de contrôler ces interfaces graphiques.

Ces interfaces graphiques utilisent les bibliothèques graphiques Tkinter et Tix pour créer l'interface graphique de l'application.

Certaines classes de cette partie ne possèdent pas de contrôleur, par leur simplicité, il ne semblait pas nécessaire de faire une classe contrôleur.

II- Bibliothèques et dépendances

Le projet est développé avec Python (version 3.6) et dépend de plusieurs bibliothèques :

- Networkx v2.4 pour la gestion des graphes
- Tkinter et Tix pour l'interface graphique
- Plotly v4.5.0 pour la visualisation 3D
- Nilearn v0.6.1 pour les données de neuroimagerie
- Pywebview v3.2 pour l'affichage de la fenêtre de visualisation

III- Connectomes de tests et graphes générés

Pour tester l'application et préparer des données pertinentes, il a fallu construire des graphes de connectomes à partir de données d'entrée (matrice d'adjacence et atlas). On a pour cela utilisé les données suivantes:

- Les fichiers matConn.txt et liste_regions_valides.txt qui sont des fichiers fournis par l'hôpital de Tours, l'atlas ne contenant pas de coordonnées pour les sommets, seulement un index et un label.

- Brodmann82.edge et Brodmann82.node qui sont la matrice d'adjacence et l'atlas Brodmann82, un connectome de 82 sommets qui est récupéré des connectomes de tests de Brain Net Viewer (https://www.nitrc.org/projects/bnv/). Ce format est un format qui a une syntaxe imposée de 6 champs pour l'atlas comprenant les coordonnées, deux valeurs numériques (qui peuvent être variables) et un label.

- L'atlas HarvardOxford.xml et la matrice d'adjacence harvardoxford_matrix.xml récupérés de FSL (https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/). Il permet de tester la lecture d'atlas en XML.

- L'atlas msdl_rois_labels.csv et msdl_matrix.txt aussi extraits de FSL (https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/) . L'atlas msdl_rois_labels.csv possède un header et peut donc servir à tester la lecture en csv et quand les champs sont définis.

Ces différents formats permettent de couvrir la majorité des types de formats utilisés en connectomique, et permettent de créer des graphes aux caractéristiques différentes: avec ou sans coordonnées, avec des attributs différents, de tailles différentes.

Des graphes de connectomes déjà construits ont aussi été utilisés, pour tester les modifications de graphe, le chargement d'un graphe déjà écrit, et pour avoir des jeux de données pertinents pour les comparaisons.

Les graphes générés lors des recherches suivantes:

- Csaba Kerepesi, Balázs Szalkai, Bálint Varga, Vince Grolmusz: The braingraph.org Database of High Resolution Structural Connectomes and the Brain Graph Tools, Cognitive Neurodynamics Vol. 11 No. 5, pp. 483-486 (2017) http://dx.doi.org/10.1007/s11571-017-9445-1 .

- Csaba Kerepesi, Balázs Szalkai, Bálint Varga, Vince Grolmusz: How to Direct the Edges of the Connectomes: Dynamics of the Consensus Connectomes and the Development of the Connections in the Human Brain, PLoS One 11(6): e0158680. http://dx.doi.org/10.1371/journal.pone.0158680, June 30, 2016.

- Balázs Szalkai, Csaba Kerepesi, Bálint Varga, Vince Grolmusz: High-Resolution Directed Human Connectomes and the Consensus Connectome Dynamics, PLOS ONE, Vol. 14 No. 4,: e0215473 (2019) https://doi.org/10.1371/journal.pone.0215473

On utilise ici le jeu de données de 1064 cerveaux avec 86 sommets récupéré sur Braingraph.org (http://braingraph.org/cms/download-pit-group-connectomes/).

- Des graphes de connectomes de rats et de vers récupérés sur Neurodata.io (https://neurodata.io/project/connectomes/).

IV- Fonctionnalités incomplètes et futures

Certaines classes possèdent des fonctions incomplètes ou vides, et qui seront donc à compléter. Parmi ces fonctions, certaines sont fonctionnelles, mais implémentée de manière simpliste et pourraient être améliorées ou remplacées.

Les fonctionnalités de comparaison ne sont pas ou peu implémentée. Une simple classe de comparaison a été développée, nommée SymmetricDifference, qui utilise l'implémentation networkx de l'algorithme éponyme pour effectuer une comparaison entre deux graphes.

De plus, toutes les fonctions liées à la comparaison dans les autres classes ne sont pas écrites (la classe CompareWindow, la fonction de comparaison de MainWindow, la classe GraphComparer qui est incomplète).

D'autres fonctionnalités peuvent être implémentée de manière plus élégante et plus extensive, comme par exemple les fonctions de visualisation ou de création de sous graphe. Actuellement, il est possible de visualiser ou construire un sous graphe représentant l'hémisphère gauche ou droit du connectome, construit simplement en séparant les nœuds par leur coordonnée X. Une amélioration de cette séparation pourrait être de chercher un attribut des nœuds démontrant l'appartenance à un des hémisphères, en s'inspirant de la recherche des attributs de la fonction de visualisation.

Toujours dans la visualisation, une autre fonctionnalité qui est incomplète est la visualisation avec un volume de cerveau entourant le graphe. Une piste a été explorée à l'aide des bibliothèques plotly et nilearn pour récupérer les coordonnées d'un mesh 3D de cerveau, mais l'exécution ne fonctionne pas.

Les fonctionnalités d'ajout d'attributs calculés au graphe sont aussi améliorables : plusieurs fonctions/attributs ont été écrites, concernant des informations pertinentes à l'étude de connectomes, mais d'autres fonctionnalités pourraient être éventuellement ajoutées.

V- Continuation du projet

Comme mentionné plus haut, une partie du projet a été modélisée, mais les classes sont restées vides par manque de temps ou priorisation d'autres fonctionnalités.

Il est possible de continuer le projet dans plusieurs directions une fois que les fonctionnalités mentionnées plus haut seront terminées.

Le modèle d'étude d'un connectome dépasse le cadre actuel de l'application : l'application gère actuellement un format d'entrée du type matrice/atlas ou graphe et crée des objets graphes à manipuler (ajouter des calculs, comparer), mais l'étude des connectomes commence au niveau des images de neuroimagerie médicale (fichiers Niftii par exemple).

Il est donc possible d'étendre le projet dans les deux directions :

- Permettre plus de fonctionnalités avec les graphes de connectomes avec plus d'algorithmes de comparaison, plus d'opération sur les graphes
- Permettre un format d'entrée avec des données de neuroimagerie et d'atlas depuis lesquelles on peut récupérer une matrice d'adjacence. On peut pour cela utiliser les bibliothèques nilearn, nibabel et nypipe