

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

[polytech.univ-tours.fr](http://polytech.univ-tours.fr)

## Projet Recherche & Développement 2019-2020

# Visualisation et comparaison de connectomes à l'aide de graphes

**Tuteurs académiques**

Nicolas MONMARCHÉ

Kieu Diem Ho

Jean-Yves RAMEL

**Étudiant**

Clément CONDETTE (DI5)



# Liste des intervenants

Nom	Email	Qualité
Clément CONDETTE	<a href="mailto:clement.condette@etu.univ-tours.fr">clement.condette@etu.univ-tours.fr</a>	Étudiant DI5
Nicolas MONMARCHÉ	<a href="mailto:nicolas.monmarché@univ-tours.fr">nicolas.monmarché@univ-tours.fr</a>	Tuteur académique, Département Informatique
Kieu Diem Ho	<a href="mailto:kieudiem.ho@univ-tours.fr">kieudiem.ho@univ-tours.fr</a>	Tuteur académique, Département Informatique
Jean-Yves RAMEL	<a href="mailto:jean-yves.ramel@univ-tours.fr">jean-yves.ramel@univ-tours.fr</a>	Tuteur académique, Département Informatique



# Avertissement

Ce document a été rédigé par Clément Condette susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Nicolas Monmarché, Kieu Diem Ho et Jean-Yves Ramel susnommés les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



## Pour citer ce document

Clément Condette, *Visualisation et comparaison de connectomes à l'aide de graphes*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2019-2020.

```
@mastersthesis{
  author={Condette, Clément},
  title={Visualisation et comparaison de connectomes à l'aide de graphes},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2019-2020}
}
```

# Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	vi
<b>I Rapport de recherche et développement</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
1 Acteurs, enjeux et contexte .....	2
2 Objectifs .....	2
3 Hypothèses .....	3
4 Bases méthodologiques .....	3
<b>2 Description générale</b>	<b>5</b>
1 Environnement du projet .....	5
2 Caractéristiques des utilisateurs .....	5
3 Fonctionnalités du système .....	6
4 Structure générale du système .....	6
<b>3 État de l'art</b>	<b>8</b>
1 Connectomes et connectomique .....	8
1.1 Apparition du terme connectome .....	8

1.2	Importance de la connectomique .....	8
2	Construction de connectomes .....	9
2.1	Connectomes fonctionnels et structurels .....	9
2.2	Connectome depuis un fichier de neuroimagerie .....	9
3	Extraction de données de connectomes .....	9
4	Connectomes et graphes .....	10
4.1	Considérer les connectomes comme des graphes .....	10
4.2	De connectome à graphe .....	10
4.3	Les caractéristiques d'un graphe de cerveau .....	10
5	Travaux sur les connectomes .....	11
5.1	Connectomes structurels .....	11
5.2	Connectomes fonctionnels .....	11
5.3	Etudes sur des pathologies .....	11
6	Outils pour l'étude de connectomes .....	12
7	Comparaison de graphes .....	12
<b>4</b>	<b>Analyse et conception</b> .....	<b>13</b>
1	Données d'entrée de l'application .....	13
1.1	Matrice d'adjacence d'un connectome .....	13
1.2	Le format NiFTI .....	14
1.3	Les atlas .....	14
1.4	Les fichiers graphes .....	15
2	Modélisation logicielle .....	15
2.1	Modèle .....	17
2.2	Vue .....	17
2.3	Contrôleur .....	17
2.4	Classes à la fin du projet .....	18
<b>5</b>	<b>Mise en œuvre</b> .....	<b>19</b>
1	Outils et librairie utilisées .....	19
2	Implémentation .....	20
2.1	Limites et problèmes rencontrés .....	20
2.2	Choix d'implémentation .....	20
<b>6</b>	<b>Conclusion</b> .....	<b>22</b>
1	Bilan du semestre 9 .....	22
1.1	Tâches effectuées au S9 .....	22
1.2	Tâches en retard au S9 .....	22
1.3	A faire au S10 .....	23
2	Bilan du semestre 10 .....	23

2.1	Tâches effectuées au S10.....	23
2.2	Reste à faire et retards .....	24
2.3	Bilan sur la qualité.....	24
2.4	Bilan sur la gestion du projet .....	25
<b>II</b>	<b>Documents annexes</b>	<b>26</b>
<b>7</b>	<b>Description des interfaces externes du logiciel</b>	<b>27</b>
1	Interfaces matériel/logiciel .....	27
2	Interfaces homme/machine .....	27
3	Interfaces logiciel/logiciel .....	27
<b>8</b>	<b>Spécifications fonctionnelles</b>	<b>31</b>
1	Définition des fonctions .....	31
1.1	Définition de la fonction 1.....	31
1.2	Définition de la fonction 2.....	31
1.3	Définition de la fonction 3.....	32
1.4	Définition de la fonction 4.....	32
1.5	Définition de la fonction 5.....	33
1.6	Définition de la fonction 6.....	33
<b>9</b>	<b>Spécifications non fonctionnelles</b>	<b>34</b>
1	Contraintes de développement et conception .....	34
2	Contraintes de fonctionnement et d'exploitation .....	34
2.1	Performances.....	34
2.2	Capacités.....	34
2.3	Contrôlabilité.....	35
2.4	Sécurité .....	35
2.5	Intégrité.....	35
<b>10</b>	<b>Livrables et conduite de test</b>	<b>36</b>
1	Gestion des livrables au cours de la phase de développement.....	36
2	Plan de conduite de test .....	37
3	Condition de validation de test .....	37
<b>11</b>	<b>Cahier de développeur</b>	<b>39</b>
1	Diagramme de classe et explications .....	39
1.1	Module Connectome Manipulation .....	39
1.2	Module Application.....	40
2	Bibliothèques et dépendances.....	40

3	Connectomes de tests et graphes générés .....	40
4	Fonctionnalités incomplètes et futures .....	41
5	Continuation du projet .....	42
<b>12</b>	<b>Gestion de projet</b> .....	<b>43</b>
1	Planification du projet .....	43
2	Cycle de développement .....	43
<b>13</b>	<b>Documentation d'installation</b> .....	<b>45</b>
1	Environnement de développement du projet .....	45
2	Exécutable et PyInstaller .....	45
<b>14</b>	<b>Documentation d'utilisation</b> .....	<b>46</b>
1	Présentation de l'application .....	46
2	Utilisation de l'application .....	47
2.1	Créer un graphe de connectome .....	47
2.2	Importer un graphe préexistant .....	51
2.3	Modifier les informations du graphe .....	52
2.4	Visualiser le connectome .....	52
2.5	Sauvegarder des graphes de connectomes .....	53
3	Bibliothèque de manipulation de connectomes .....	53
3.1	ConnectomeObject .....	54
3.2	ConnectomeGraph .....	54
<b>15</b>	<b>Cahier de tests</b> .....	<b>55</b>
1	Tests de graphes .....	55
1.1	Test hospital .....	55
1.2	Test edge node .....	55
1.3	Test xml .....	56
1.4	Test csv .....	56
1.5	Test from graph brodmann .....	56
1.6	Test from graph hospital .....	56
1.7	Test from graph harvardoxford .....	57
1.8	Test from graph msdl .....	57
1.9	Test calculations .....	57
1.10	Test Subgraph .....	57
1.11	Test SymmetricDifference .....	58
2	Tests de visualisation .....	58
2.1	Test brodmann82view .....	58
2.2	Test hospitalview .....	58



2.3	Test msdlview .....	59
2.4	Test harvardoxfordview .....	59
2.5	Test visualizeLeft .....	59
2.6	Test visualizeRight.....	59
<b>Webographie</b>		<b>60</b>
<b>Bibliographie</b>		<b>61</b>
<b>Glossaire</b>		<b>63</b>

# Table des figures

## 2 Description générale

- 1 Diagramme de cas d'utilisation de l'application ..... 6
- 2 Diagramme de composants de l'application ..... 7

## 4 Analyse et conception

- 1 Diagramme de classe de l'application..... 16
- 2 Diagramme de classe de l'application après révision ..... 16
- 3 Diagramme de classe final ..... 18

## 6 Conclusion

- 1 Trello des tâches de la phase de développement ..... 25

## 7 Description des interfaces externes du logiciel

- 1 Mock-up de la fenêtre principale de l'application..... 28
- 2 Mock-up de la fenêtre d'import de données ..... 28
- 3 Mock-up de la fenêtre de visualisation de connectomes ..... 29
- 4 Mock-up de la fenêtre de visualisation de graphe ..... 30

## 10 Livrables et conduite de test

- 1 Diagramme de Gantt de la phase de développement..... 36

## 11 Cahier de développeur

- 1 Diagramme de classe final du projet de recherche et développement..... 39

**12 Gestion de projet**

1	Diagramme de Gantt du projet de recherche et développement .....	43
2	Diagramme de Gantt final du développement .....	44

**14 Documentation d'utilisation**

1	Fenêtre principale de l'application .....	47
2	Menu fichier de l'application .....	48
3	Fenêtre de création de connectome.....	48
4	Extrait de l'atlas msdl rois.csv.....	49
5	Extrait de la matrice Brodmann82.edge .....	49
6	Options de constructions de connectome .....	50
7	Connectome ajouté à la liste.....	51
8	Import d'un graphe graphml .....	51
9	Affichage des informations sur le graphe.....	52
10	Visualisation d'un graphe de connectome.....	53

Première partie

# Rapport de recherche et développement

# 1

## Introduction

Dans ce chapitre, on décrit le contexte du projet et la structure du rapport ainsi que les acteurs, les objectifs, l'hypothèses, et les bases méthodologiques sur la gestion de projet.

### 1 Acteurs, enjeux et contexte

Au cours des dernières années, de nombreux scientifiques se sont penchés sur l'idée de modéliser le cerveau humain à l'aide de connectomes, une carte des connexions entre les différentes zones du cerveau. L'utilisation de connectomes pourrait aider à faire progresser la médecine en permettant de pouvoir étudier la manière dont interagissent entre elles les différentes zones du cerveau, ou comparer les cerveaux de différents patients et pouvoir tirer des conclusions des différences trouvées. Cela permettrait d'avoir une meilleure compréhension du fonctionnement du cerveau dans le cas de certaines pathologies telles que la maladie d'Alzheimer [7].

La connectomique, qui est l'étude des connectomes, utilise les graphes pour représenter les connectomes et pouvoir les étudier. En effet, l'utilisation de graphes est très utile pour représenter les connectomes, les graphes de connectomes étant une représentation simple d'un objet complexe tel que le cerveau tout en permettant d'appliquer la théorie des graphes pour pouvoir étudier les connectomes.

Il existe déjà des algorithmes de comparaisons de graphes qui pourraient être utiles à l'étude des connectomes, mais il existe un manque d'outils permettant de modéliser les connectomes sous forme de graphes compatibles avec ces algorithmes. Ainsi, ce projet vise à combler ce manque en proposant une application permettant de construire des graphes à partir de données de connectomes.

Dans le cadre de ce projet, nous avons donc une répartition des rôles effectuées ainsi :

- Client : Jean-Yves Ramel
- MOA : Nicolas Monmarché et Kieu Diem Ho
- MOE : Clément Condette

### 2 Objectifs

L'objectif de ce projet de recherche est de pouvoir modéliser sous forme de graphes des connectomes pour pouvoir les rendre compatibles avec les algorithmes de comparaison précédemment

développés et pouvoir étudier les connectomes en appliquant les méthodes de la théorie des graphes.

Il va pour cela falloir dans un premier temps modéliser un connectome sous la forme d'un graphe. Le premier objectif de ce projet est donc de développer une méthode pour modéliser sous forme de graphe des données de connectomes. On va pour cela utiliser Python pour créer des fichiers permettant de représenter des graphes compatibles avec les paramètres d'entrée des algorithmes de comparaison.

Une fois un connectome transformé en graphe, il est nécessaire d'avoir une représentation graphique du graphe pour pouvoir l'étudier et vérifier s'il est correctement construit. Il est donc nécessaire de développer un module pour visualiser le graphe. Ainsi, l'application à développer au cours de ce projet doit pouvoir répondre à plusieurs besoins :

- Une méthode d'import des données à étudier pour pouvoir les modéliser sous forme de graphe
- Une transformation des données de connectomes vers un graphe compatible avec l'[Ant Colony Optimization](#)
- Une méthode pour visualiser les connectomes sous forme de graphe
- L'intégration de l'algorithme de comparaison de graphe

Le développement de l'application impliquera donc des modules pour réaliser ces différents objectifs.

### 3 Hypothèses

Pour ce projet, il est nécessaire de développer un outil permettant de visualiser le connectome étudié sous forme de graphe.

Ayant au cours des recherches eu accès à un échantillon de données de l'hôpital de Tours, on suppose que les données d'entrée de l'application seront sous le même format : un fichier contenant les informations sur les sommets(l'atlas), et une matrice d'adjacence permettant de définir les arêtes. Les deux fichiers étaient au format .txt dans l'échantillon fourni.

La visualisation du connectome sous forme de graphe peut être faite dans un navigateur pour utiliser la vue interactive de Nilearn.

Si le projet progresse rapidement et que le temps est disponible, on pourrait proposer une possible évolution de l'application en ajoutant des options à l'utilisateur lors de la construction de graphes. Ces options permettraient d'ajouter des valeurs aux sommets aux arêtes telles que le degré des sommets, l'excentricité d'un sommet, le poids des arêtes ou la longueur des arêtes par exemple.

Pour la comparaison de graphe, l'algorithme utilise une variante de l'[Ant Colony Optimization](#) pour résoudre le problème. Si le temps le permet, plus d'algorithmes de comparaison de graphes pourront être ajoutés.

### 4 Bases méthodologiques

Dans le cadre de ce projet, le travail de développement sera effectué dans le langage Python, et ce pour plusieurs raisons :

- Le langage Python est un des langages privilégiés par le client.
- Le langage Python possède des outils et bibliothèques propices à l'étude de données de neuroimagerie et de graphes
- Cela permet une intégration facile de l'algorithme de comparaison de graphes déjà existant.

Ensuite, pour traiter des données de neuroimagerie, on va utiliser des bibliothèques comme Nilearn, Nipype et Nibabel, et créer des fichiers de graphes à l'aide de NetworkX.

Enfin, pour représenter les fonctionnalités du système, nous allons utiliser le langage UML.

Pour la gestion du cycle de développement, le client étant physiquement présent dans l'enceinte de Polytech Tours, un contact régulier pour un feedback est possible. Le déroulement du projet se fera donc en méthode agile, plus particulièrement en méthode SCRUM.

Il va ainsi être possible de réaliser des sprints d'une ou deux semaines et d'interagir avec le client pour vérifier que le produit ne dérive pas du produit attendu.

# 2

## Description générale

### 1 Environnement du projet

Pour ce projet, en plus de la partie représentation de connectomes sous forme de graphe, il existe une partie centrée sur la comparaison de graphes. Pour ce problème, nous allons utiliser le « multivalent graph matching » pour la comparaison des graphes modélisés. Après cela, on applique une variante de l'[Ant Colony Optimization](#) pour résoudre ce problème. L'algorithme de comparaison de graphe avec l'[Ant Colony Optimization](#) a déjà été développé par Diem.

L'application prend en entrée des connectomes générés à partir de données de neuroimagerie. Pour cela, la génération du connectome se fera en amont du projet avec des logiciels tiers pour générer des données d'entrées dans un format compatible avec l'application. Les données d'entrée attendues seront expliquées plus en détail dans la partie spécifications fonctionnelles.

### 2 Caractéristiques des utilisateurs

L'application est dans un premier temps destinée à être utilisée dans le cadre de la recherche, on suppose cependant que cette application sera ensuite utilisée par du personnel médical. Il existe donc deux types d'utilisateurs principaux pour cette application :

- Le type d'utilisateur principal de l'application sera un chercheur travaillant sur les connectomes :
- Il a des connaissances en informatique autre que celles nécessaires à l'utilisation de cet outil.
  - Il sait utiliser correctement l'application (paramètres, choix du fichier, interprétation du résultat).
  - Il est un utilisateur régulier de l'application pour effectuer des recherches.
  - Il possède tous les droits. Un second utilisateur de l'application serait un personnel médical :
  - Il ne possède à priori pas de connaissances en informatique.
  - Il sait utiliser correctement l'application (paramètres, choix du fichier, interprétation du résultat).
  - Il est un utilisateur occasionnel de l'application dans un cadre médical.
  - Il possède tous les droits.



### 3 Fonctionnalités du système

L'objectif du projet est de faire une application capable de comparer des graphes représentant des connectomes humain pour analyser les différences entre patients. Il est donc nécessaire pour remplir ces objectifs qu'il existe à minima les fonctionnalités suivantes :

- Importer des fichiers permettant de générer une représentation sous forme de graphe de connectomes.
- Générer un graphe contenant les informations d'un ou plusieurs connectomes.
- Proposer la comparaison de ce graphe avec un ou plusieurs autre graphes (soit un graphe importé ou un graphe de référence).
- Visualiser un connectome importé.
- Visualiser un graphe sélectionné. Le projet porte plus d'importance aux fonctionnalités de l'application qu'à son ergonomie, l'interface graphique restera donc simpliste.

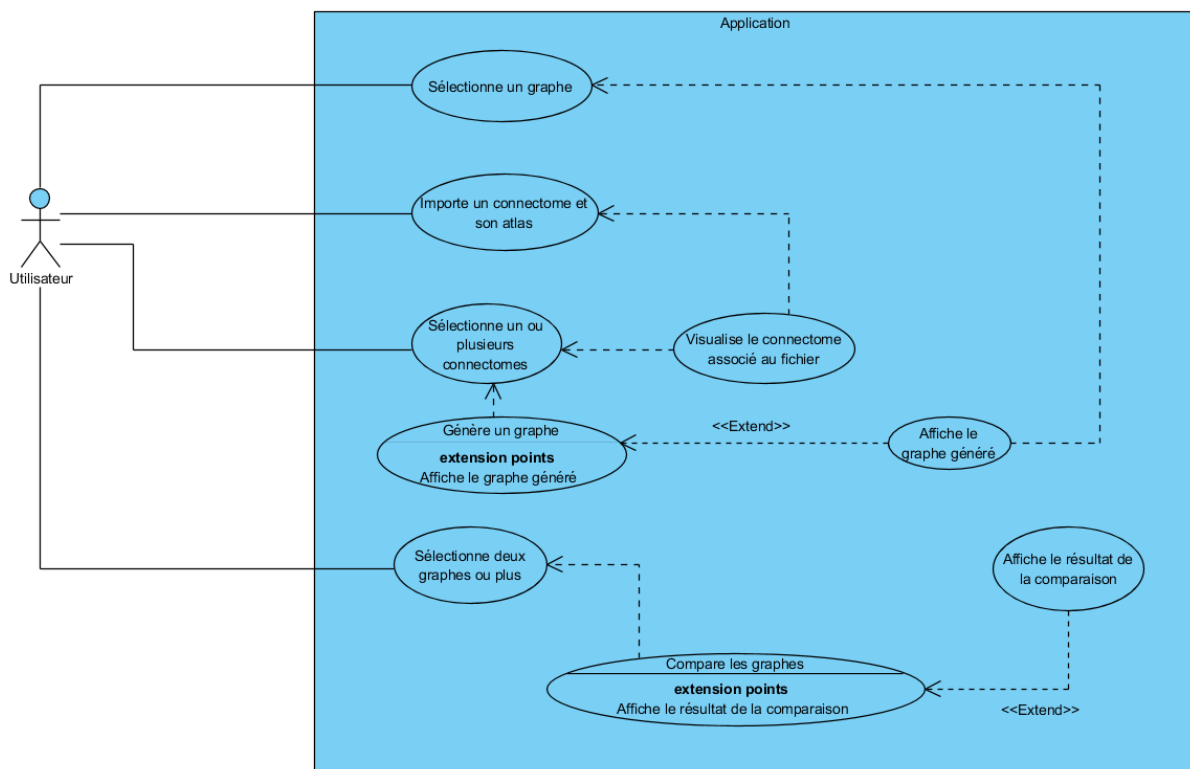


Figure 1 – Diagramme de cas d'utilisation de l'application

Le graphe généré pourra être construit à partir de plusieurs connectomes ou des données supplémentaires peuvent être ajoutées au graphe (telles que des attributs sur les arêtes ou les sommets).

A minima, un graphe construit à partir de connectome est construit à partir d'un atlas ne contenant que les noms des sommets et d'une matrice de connectivité binaire : le graphe généré aura donc des coordonnées attribuées aléatoirement pour les sommets.

### 4 Structure générale du système

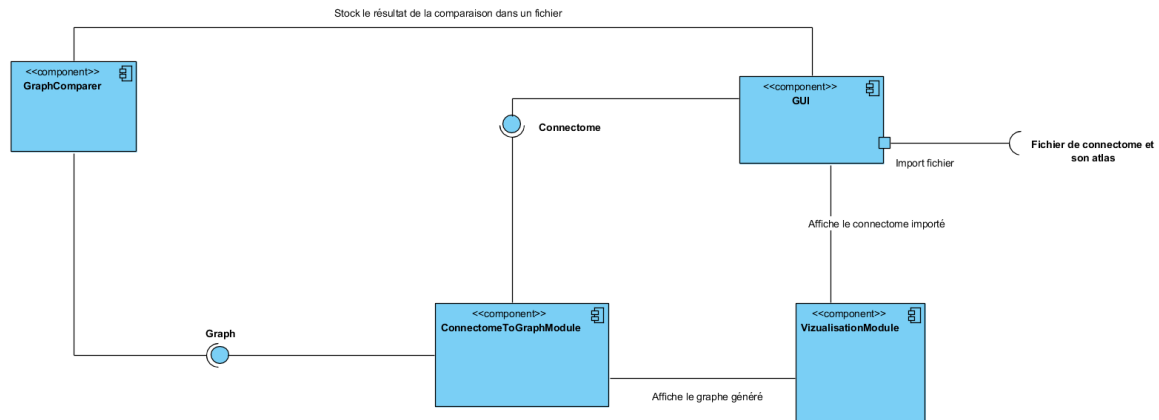
L'application doit remplir plusieurs fonctionnalités distinctes, on aura donc des composants dédiés aux différentes fonctionnalités :

- Une interface graphique qui permettra de naviguer dans les fichiers disponibles et sélectionner

les opérations à effectuer.

- Un module de calcul qui effectuera le traitement de calcul de connectomes et génération de graphe.
- Un module de visualisation de connectome sous forme de graphe et de graphe généré.
- Un module de calcul pour la comparaison de graphes.

Ci-dessous le diagramme de composant de l'application qui présente le découpage de l'application en plusieurs modules ainsi que leurs interactions.



**Figure 2 –** Diagramme de composants de l'application

L'interface graphique fera ici office de vue et permettra à l'utilisateur de sélectionner les actions à effectuer, et permettra de sélectionner les entrées à fournir aux autres modules :

- Le module de visualisation prendra un objet représentant le graphe généré dans le cadre de l'affichage d'un graphe, ou deux fichiers contenant la matrice d'adjacence et l'atlas d'un connectome pour l'affichage d'un connectome.
- Le module de comparaison prendra au moins un objet graphe, qui sera ensuite comparé à un graphe de référence si un seul graphe est sélectionné, ou à d'autres graphes sélectionnés.
- Le module de modélisation sous forme de graphe prend un ou plusieurs fichiers de matrice d'adjacence de connectomes avec l'atlas correspondant.

# 3

## État de l'art

### 1 Connectomes et connectomique

#### 1.1 Apparition du terme connectome

Le terme [Connectome](#) a été utilisé pour la première fois en 2005 par Dr. Olaf Sporns [6] et Dr. Patric Hagmann [8] pour parler d'une carte complète des connexions du cerveau.

Le terme connectome est aujourd'hui utilisé pour parler d'une carte des connexions allant de la carte détaillée à l'échelle cellulaire du système neuronal jusqu'à une représentation à macro échelle pour identifier la connectivité entre les régions du cerveau.

#### 1.2 Importance de la connectomique

L'étude des connectomes est la **connectomique**, l'analyse de la connectivité du cerveau. A l'aide de connectomes, les scientifiques peuvent mieux comprendre le fonctionnement du cerveau et, en effectuant des corrélations entre les caractéristiques des connectomes et certains paramètres d'étude, tels que le sexe du sujet [WWW6], permettre de comprendre l'influence de certains paramètres sur le cerveau et comment les différentes régions communiquent.

La connectomique permet aussi d'étudier comment les connexions du cerveau sont impactées par des pathologies telles que la maladie d'Alzheimer [7].

De son importance dans le domaine de la neuroscience, les recherches sur le connectome humain ont augmentées depuis les débuts de la connectomique, particulièrement depuis le début du [Human Connectome Project](#) en 2009. Le Human Connectome Project est un projet visant à construire une carte complète des connexions neurales structurelles et fonctionnelles du cerveau humain.

De plus, lorsque la technologie d'imagerie et de modélisation aura progressé suffisamment pour représenter entièrement un connectome humain à l'échelle cellulaire, un cerveau artificiel pourrait être construit et il est théorisé que ce cerveau artificiel aurait la même personnalité que le connectome sur lequel il est construit [WWW2].

## 2 Construction de connectomes

### 2.1 Connectomes fonctionnels et structurels

Dans le domaine de la connectomique, l'étude du cerveau peut se faire sous deux angles d'approche : une approche structurelle des connexions du cerveau ou une approche fonctionnelle.

L'approche structurelle consiste à étudier les connexions anatomiques du cerveau. Le cerveau est pour cela séparé en différentes zones anatomiques pour lesquelles on indique si ces régions sont connectées par des fibres de matière blanche. Ces mesures sont effectuées sur des [IRM](#) de diffusion. Une [IRM](#) de diffusion fonctionne en faisant apparaître la diffusion aléatoire des molécules d'eau, ce qui permet de révéler la structure des tissus du cerveau et d'effectuer des mesures de densité et d'organisation structurelle [[WWW3](#)]. Cela permet de construire un [tractogramme](#), un réseau tridimensionnel des connexions anatomiques entre les régions du cerveau.

L'approche fonctionnelle consiste à définir des points d'étude et d'étudier leurs activations en fonction du temps. On étudie pour cela une [IRM](#) fonctionnelle pour enregistrer les variations de concentration de déoxy-hémoglobine dans le sang (c'est-à-dire la concentration de saturation d'oxygène), ce qui permet de mettre en valeur les activations des différentes zones du cerveau. En effet, ces variations permettent de montrer le fonctionnement des régions cérébrales supportant la cognition, car l'augmentation du débit sanguin signale l'activation d'une région.

On effectue ces mesures sur un sujet dit "au repos", ce qui permet d'étudier des oscillations régulières du flux sanguin qui vont définir les régions fonctionnellement connectées.

### 2.2 Connectome depuis un fichier de neuroimagerie

Il existe plusieurs modèles de représentation de données de neuroimagerie, et un des plus couramment utilisé est le format [NiFTI](#). Ce format de fichier contient des images d'[IRM](#) à partir desquelles on va pouvoir récupérer les informations sur les connexions du cerveau. Dans le cadre des connectomes, nous utilisons ici une méthode permettant d'extraire les connexions entre les [Regions Of Interest](#) définie par un atlas donné.

Ainsi, à l'aide de l'atlas il est possible de définir les points d'intérêt qui représentent les sommets du connectome, et il va pouvoir être possible d'étudier les connexions entre ces différentes régions du cerveau.

## 3 Extraction de données de connectomes

Dans le domaine de la recherche, les connectomes sont principalement utilisés comme un outil pour représenter les connexions cérébrales et faciliter l'extraction d'information. Le type d'information exploitable dans un connectome fonctionnel sont les informations sur l'intensité et la présence de connexions entre différentes zones du cerveau. Une fois ces données extraites, les différentes recherches permettent d'inférer certains principes ou d'établir la corrélation entre certaines caractéristiques du connectome et des éléments médicaux ou sociaux tels que certaines pathologies ou des critères tels que le sexe, l'âge, l'état de santé, le stress etc.

L'extraction de données d'un connectome se fait en exploitant les valeurs de la matrice de connectivité du connectome : on applique aux poids des connexions entre différents sommets une régression linéaire sur les différentes caractéristiques à étudier, ce qui permet de créer une statistique T avec laquelle on peut construire une matrice T de résultat qui permet d'étudier l'impact des caractéristiques choisies [[WWW6](#)].

## 4 Connectomes et graphes

### 4.1 Considérer les connectomes comme des graphes

Quand on parle de travaux sur les connectomes, on finit inévitablement par parler de théorie des graphes. En effet, en neuroscience la théorie des graphes est une approche très populaire pour analyser les différentes caractéristiques du modèle.

Dans le cas de la connectomique, le modèle est le connectome : une construction composée de régions d'intérêts et de connexions entre ces régions, une modèle qui est facilement modélisable à l'aide d'un graphe. Les régions représentent les sommets du graphes et les connexions les arêtes.

On peut ensuite appliquer des méthodes de théorie des graphes pour récupérer des informations sur le réseau, telles que le degré d'un sommet ou la force des connexions entre des sommets clés.

Un connectome est fondamentalement un réseau composés de régions en tant que sommets connectés. Ainsi, les connectomes peuvent être considérés comme des graphes pour lesquelles on peut construire une matrice d'adjacence.

### 4.2 De connectome à graphe

La construction d'un graphe de connectomes nécessite l'utilisation de deux composants : la matrice d'adjacence et l'atlas.

L'atlas va permettre de définir les informations sur les sommets du connectome : leur nombre, leur label et éventuellement leurs coordonnées.

La matrice d'adjacence va contenir les informations sur les arêtes : en effet, les lignes et colonnes de la matrice d'adjacence représentent les sommets, et les valeurs de la matrice représentent les arêtes entre ces sommets [WWW1]. Les valeurs peuvent dénoter la présence d'une arête, mais aussi leur poids dans le cas d'un graphe pondéré, ou encore leur direction dans le cas d'un graphe orienté. Il est cependant possible de transformer le graphe en graphe binaire ou non-orienté à l'aide d'opérations de symétrisation et de seuillage si nécessaire.

Une opération de symétrisation vise à binariser la matrice en utilisant sa transposée pour retire les informations de direction.

Une opération de seuillage sert à retirer les poids des arêtes en définissant une valeur seuil au dessus de laquelle on garde les arêtes, et en dessous de laquelle on ignore les arêtes.

Les arêtes représenteront des valeurs différentes dans le cas d'un connectome fonctionnel ou structural : un connectome fonctionnel représentera les taux d'activation de connexion entre les régions là où un connectome structural cherchera à montrer les connexions anatomique telles que la concentration de matière blanche.

Des graphes complets ont déjà été recensé pour des espèces autres que l'Homme, tel que le ver *Caenorhabditis elegans* [9] ou d'autres espèces animales [WWW4].

### 4.3 Les caractéristiques d'un graphe de cerveau

Les graphes construits à partir de connectomes ont certaines caractéristiques particulières.

Le graphe de cerveau est un réseau "**petit monde**", c'est-à-dire que pour un sommet A donné, même s'il n'existe pas de connexions directe avec un sommet B, il y a de grandes chances qu'un des voisins de A ait une connexion à B et qu'on ait donc des chemins court pour aller de A à B.

De nombreuses mesures de théorie des graphes sont applicables et intéressantes pour étudier le cerveau. Dans le cas de la connectomique, on s'intéresse particulièrement à des métriques telles que le degré des sommets ou l'excentricité globale ou le poids moyen des arêtes pour pouvoir définir la force des connexions dans le cerveau.

## 5 Travaux sur les connectomes

De nombreuses études ont été faites sur les connectomes dans les dernières années. La plupart de ces études se concentrent sur l'analyse de connectomes pour étudier l'impact de certaines caractéristiques ou pathologies. Des études ont par exemple été conduites en analysant des séries de connectomes pour étudier l'impact de l'âge [WWW5] ou du sexe [WWW6].

### 5.1 Connectomes structurels

Une étude de 2014 portant sur les connectomes structurels a visé à montrer les différences entre les connectomes chez l'homme et la femme [WWW6]. Cette étude a permis de montrer que les connexions chez la femme étaient plus fortes que chez l'homme dans le cas de connexions inter-hémisphère là ou chez l'homme, les connexions intra-hémisphères sont plus fortes.

Une autre étude de 2014 a tenté d'examiner des connectomes structurels pour trouver s'il existe une corrélation entre une structure anormale du connectome et le syndrome de dépression. Cette étude, basée sur un atlas composé de 84 régions, montre que les connexions de matière blanche dans le cortex frontal et l'hypothalamus sont plus faible chez les gens atteints de dépression et qu'il y a donc un impact anatomique à la dépression.

### 5.2 Connectomes fonctionnels

L'organisation fonctionnelle du cerveau varie entre les individus. Une étude conduite en 2015 [2] vise à démontrer qu'un profil de connectivité permet d'identifier un individu au sein d'un large groupe au même titre que des empreintes digitales.

En utilisant les matrices d'adjacence, une identification a été faite pour associer plusieurs séries de matrices d'adjacence à différents états : au repos et pendant des activités, et des comparaisons entre ces séries ont été effectuées pour apparier ces matrices. Les résultats ont montré que les différents états appariés était très souvent du même sujet et qu'il existe donc des caractéristiques montrant l'unicité du connectome.

En 2017, une étude a utilisé des graphes de connectomes pour mettre en place un protocole de prédiction de modèle de comportement du cerveau en utilisant les données du Human Connectome Project [4]. Cette étude a montré une forte corrélation entre la connectivité fonctionnelle du cerveau et le niveau d'intelligence ou la rétention de l'attention.

### 5.3 Etudes sur des pathologies

Des études sont aussi faites pour examiner l'influence de certaines pathologies sur les connexions cérébrales comme la maladie d'Alzheimer [7], [5] ou l'étude de l'autisme [1].

L'étude sur la maladie de Parkinson avait pour but de comparer des sujets atteints de Parkinson et de sujets atteints de Parkinson et de troubles cognitifs légers pour comparer les altérations du connectome. Le résultat montre que l'altération du réseau neuronale résulte dans des déficits cognitifs.

Le fait que la maladie d'Alzheimer endommage les connexions du cerveau est un fait connu depuis des années, mais ces études ont cherché, en effectuant des recherches sur les connexions à la macro-échelle, de voir les impacts sur les connexions chez les sujets à risque d'Alzheimer. On a pu voir que le connectome fonctionnel chez ces sujets avait en général une connectivité réduite et une longueur de chemin caractéristique plus longue.

Une autre étude de 2017 portait sur l'étude du spectre de l'autisme [1] visant à montrer les différences de connexions lo long du spectre.

## 6 Outils pour l'étude de connectomes

Il existe beaucoup d'outils pour l'étude de la neuroscience et donc des connectomes, ces outils allant du traitement de neuroimagerie à la modélisation et visualisation de connectomes.

Des outils tels que BrainNetViewer, Trackvis ou Connectome Workbench permettent la visualisation de connectomes fonctionnels et structuels.

D'autres outils permettent d'effectuer des traitements sur les données d'IRM, en corrigeant les valeurs où en isolant certaines valeurs qui nous intéressent. on peut utiliser pour cela des outils comme Freesurfer, MRtrix ou Diffusion Toolkit.

## 7 Comparaison de graphes

Les méthodes de comparaison de graphe ont beaucoup évolué au fil du temps et de nombreuses méthodes existes. On va dans notre cas s'intéresser à la comparaison de graphes multi-labelés avec l'algorithme de la colonie de fourmis [Ant Colony Optimization](#). La description du fonctionnement de l'algorithme de l'[Ant Colony Optimization](#) et de la comparaison de graphe utilisant cet algorithme est traduit de l'anglais depuis le document [3].

La comparaison de graphes n'étant pas le point principal du projet, la description qui en sera faite n'entrera pas dans les détails du fonctionnement de l'algorithme.

L'[Ant Colony Optimization](#) est une approche bio-inspirée qui sert à résoudre des problèmes d'optimisation. L'idée principale est de résoudre le problème en cherchant le chemin au coût le plus faible dans un graphe et d'utiliser des fourmis artificielles pour chercher les meilleurs chemins.

Le comportement des fourmis est le inspiré du comportement de vraies fourmis : elles déposent des trainées de phéromones sur les arêtes et choisissent leur chemin en fonction des phéromones précédemment déposées.

# 4

## Analyse et conception

Dans cette partie, on va s'intéresser à la mise en place des différents modules et à la conception d'une application permettant de répondre aux besoins exprimés par le projet.

### 1 Données d'entrée de l'application

Pour fonctionner, l'application aura besoin de la matrice d'adjacence d'un connectome et de l'atlas utilisé pour sa construction.

Le format d'entrée de l'application nécessite des format de fichiers lisibles, on va donc devoir développer une fonction pour la lecture de ces données d'entrées.

Les premières pistes de développement de cette fonction ont été pensées après avoir effectué des tests sur le jeu de données de l'hôpital de Tours.

La fonction va donc être construite pour permettre de lire des fichiers textes contenant les informations à extraire pour la construction du connectome.

#### 1.1 Matrice d'adjacence d'un connectome

La matrice d'adjacence d'un connectome est construite à l'aide de données de neuroimagerie et d'un atlas pour construire les nœuds et les arêtes du réseau. Cette matrice d'adjacence va ensuite être fournie à l'application sous forme d'un fichier contenant les arêtes entre les différents nœuds.

Dans le cadre de notre application, la matrice d'adjacence est une matrice carrée à minima binaire.

Pour des soucis de lisibilité lors de la visualisation et parce que les données sont peu intéressante dans le cadre de nos études, la diagonale de la matrice n'est pas calculée.

Si la matrice d'adjacence n'est pas binaire, les valeurs permettent d'attribuer un poids aux arêtes.

On suppose que les arêtes sont dirigées si des valeurs négatives sont présentes dans la matrice d'adjacence.

Ainsi, pour la valeur aux coordonnées  $[1][2]$  de la matrice, si la valeur est positive, l'arc est dirigé vers le sommet 2, si elle est négative, l'arc est dirigé vers le sommet 1.



Avec différentes options de lectures, on peut décider de lire la matrice d'adjacence de manière non-orientée ou non-pondérée si on le nécessite, ce qui permet d'appliquer certains calculs de théorie des graphes aux graphes de connectomes.

## 1.2 Le format NiFTI

Le format de fichier **NiFTI** est un format de fichier qui a été établi il y a une dizaine d'années pour remplacer le format de fichier Analyze7.5. C'est un format adapté à la neuroimagerie.

Un fichier **NiFTI** contient :

- Des coordonnées 3D pour représenter les images de cerveau dans un environnement 3D.
- Des codes pour indiquer la signification des données
- Une méthode de stocker des valeurs sous forme de vecteurs jusqu'à 4 dimensions

A l'aide de ce format, nous allons pouvoir récupérer séries temporelles d'**IRM** fonctionnelle sur lesquelles on va pouvoir appliquer les atlas pour avoir la connectivité entre les régions. On peut donc déduire la matrice d'adjacence d'un connectome depuis ses données de neuroimagerie et son atlas.

Ces images ne sont pas utilisées dans les fonctionnalités actuelles du projet, mais pourraient servir dans des fonctionnalités futures pour permettre plus de types de données d'entrée possibles ou effectuer plus d'analyse des connectomes.

## 1.3 Les atlas

Pour générer un connectome depuis un fichier **NiFTI**, il est nécessaire de sélectionner un atlas. Un atlas est un découpage du cerveau en **Regions Of Interest** qui serviront à délimiter le cerveau en zones distinctes pour lesquelles on va pouvoir mesurer l'intensité des connexions. Chaque **Regions Of Interest** représentera un nœud avec des coordonnées 3D (x, y, z). Il existe de nombreux atlas dédiés à des représentations spécifiques du cerveau qui ont été construits en étudiant des séries de patients de tous âges et sexes.

Voici une liste non-exhaustive d'atlas existants :

- Craddock parcellation atlas (2012)
- Destrieux cortical atlas (2009)
- Harvard-Oxford atlas
- MSDL brain atlas
- Smith ICA and BrainMap atlas (2009)
- Yeo 2011 atlas
- Destrieux cortical atlas (2010)
- Schaefer parcellation (2018)

L'atlas peut être fourni en entrée par l'utilisateur ou être téléchargé depuis la source si une connexion internet est disponible.

Dans le cas où l'atlas est fourni en entrée, il peut y avoir plusieurs niveaux d'informations fournies :

- A minima, un label pour les sommets
- Des coordonnées 3D pour les sommets
- Des informations de neuroimagerie sur les sommets

Ces informations seront disposées sur une ligne par sommet, chaque information séparée par un caractère de séparation tel qu'une virgule ou une tabulation.

Un document texte sera généré au cours de la phase de préparation au développement à partir du *MSDLbrainatlas* et enregistré dans un fichier de texte qui servira de modèle aux fichiers d'atlas.

L'échantillon de l'hôpital de Tours était au format txt, on suppose donc dans un premier temps, le lecteur de données d'entrée fonctionnera pour les fichiers au format txt.

L'objectif dans un premier temps est de pouvoir lire les formats de fichiers "classiques" pour les atlas de connectomes qui sont utilisés ou générés par les autres applications d'étude de neuroimagerie. On va donc pour cela utiliser des échantillons d'atlas provenant de ces applications, comme par exemple les fichiers nodes de Brain Net Viewer ou les fichiers d'atlas csv et xml de FSL.

Les fichiers d'atlas pouvant être de beaucoup de formats différents et beaucoup de syntaxe différentes, l'objectif est de construire des fonctions de lectures capables de lire les formats classiques (xml, csv, node qui ont des syntaxes identifiables) et de s'adapter pour la lecture de formats différents pour construire au mieux les graphes correspondant. On va pour cela construire un lecteur capable d'identifier les champs de l'atlas au mieux en identifiant le type de données proposées.

Bien évidemment, cette méthode ne couvrira pas toutes les possibilités

## 1.4 Les fichiers graphes

Un autre type de fichier d'entrée nécessaire pour l'application est le type de fichier contenant un graphe de connectomes. En effet, il est intéressant pour l'application de pouvoir recharger des connectomes déjà créés précédemment, ou lire des graphes construits hors de l'application.

Il existe beaucoup de formats permettant la lecture et écriture de graphe dans des fichiers, mais le projet travaillant avec Networkx pour la modélisation des graphes, il est important de commencer avec un format de fichier supporté par Networkx.

On se tourne donc dans un premier temps vers le format graphml, qui est supporté par Networkx et qui va nous permettre de sauvegarder les graphes créés à partir de matrices et d'atlas.

Dans un second temps et si le temps le permet, le format gxl est un autre format qui serait intéressant à gérer, mais ce n'est pas un format supporté par Networkx, et l'investissement temporel pour développer cette fonctionnalité est donc plus importante.

## 2 Modélisation logicielle

Pour la réalisation de ce projet, on va utiliser le pattern MVC pour le modèle de cette application pour pouvoir mieux séparer les différents modules de l'application (visualisation, modélisation et comparaison).

Après avoir commencé le développement, une discussion par rapport au développement du projet a eu lieu pour discuter de la réalisation du projet. A la suite de cette discussion, il a été décidé que plutôt que de développer l'application seule, il serait plus intéressant de créer une librairie de manipulation de connectomes, et de créer des classes nécessaires à plus de fonctionnalités que celles nécessaires pour le projet, avec une philosophie visant à plutôt développer de manière qualitative que quantitative : toutes les classes de la librairie n'entre pas dans le cadre direct du projet et ne sont donc pas vouées à être développées directement au cours de ce projet, mais un travail de conception et modélisation préliminaire peut tout de même être effectué.

Ainsi, un second diagramme de classe a été construit.

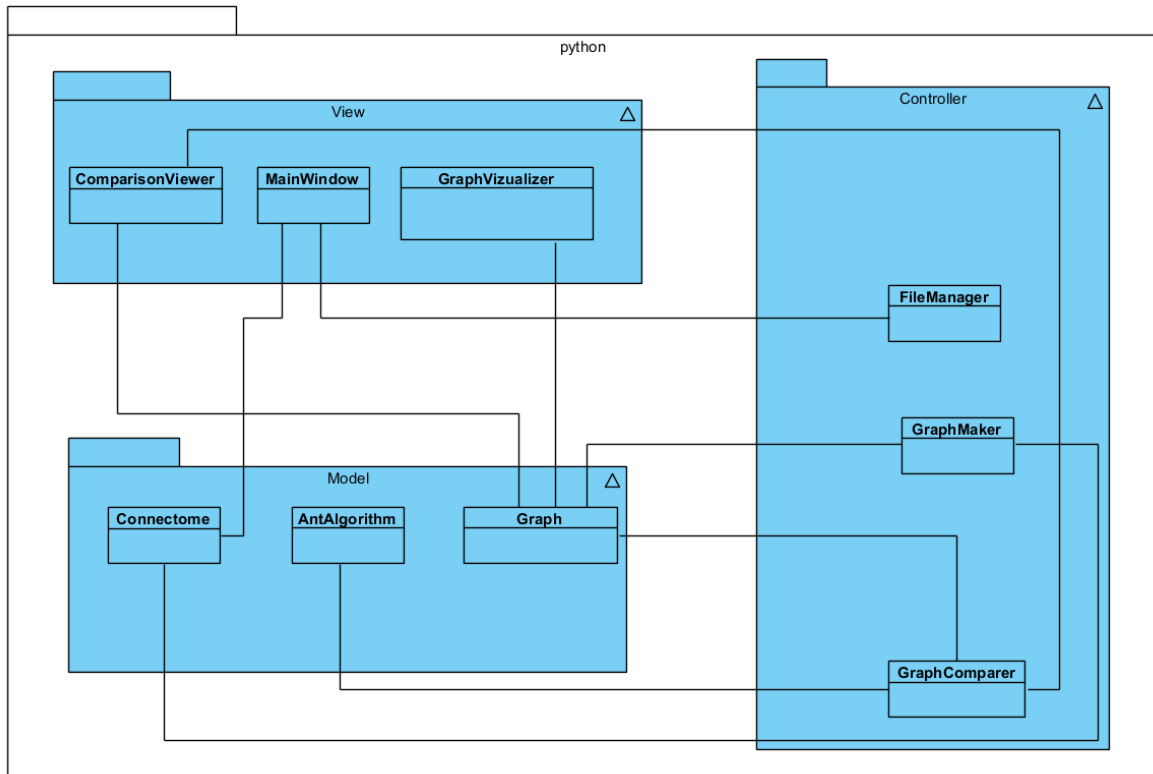


Figure 1 – Diagramme de classe de l'application

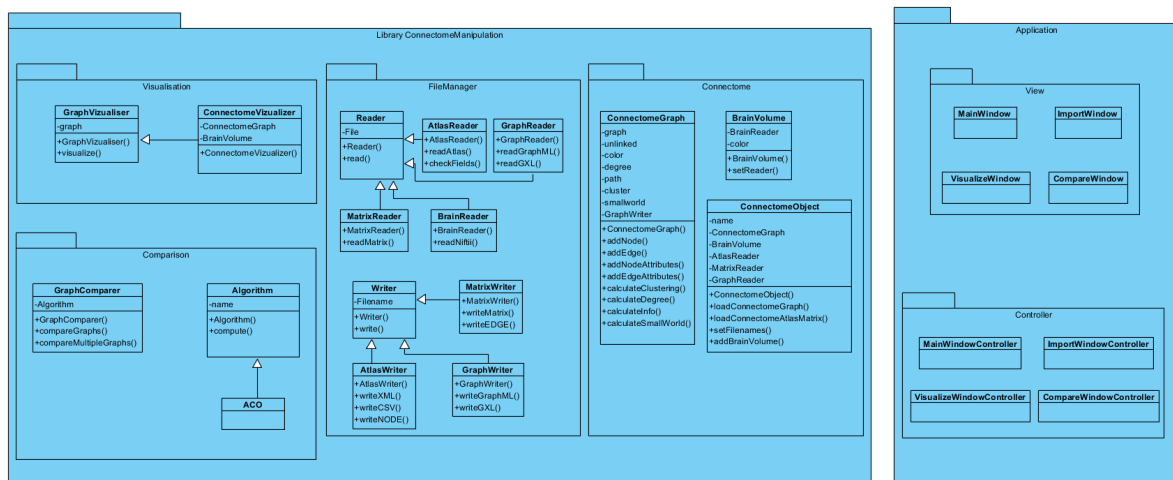


Figure 2 – Diagramme de classe de l'application après révision

Ce second diagramme propose une séparation en plusieurs modules :

- Un module Connectome qui contient les classes permettant de représenter les connectomes ou leurs éléments
- Un module FileManager qui permet de lire ou écrire des fichiers en rapport avec les connectomes
- Un module Visualization qui contient les classes en charge de la modélisation 3D des graphes de connectomes
- Un module Comparison qui permet d'utiliser des algorithmes de comparaison de graphes de connectomes
- Un module Application qui contient les classes en charge de l'IHM de l'application, ces classes suivant un modèle MVC avec des sous-modules Controller et View qui gèrent les classes Model

des modules présentés précédemment.

## 2.1 Modèle

Un premier découpage de l'application en modèle avait été fait au début du projet, et se disposait ainsi :

Pour cette application, le Modèle comportera 3 classes permettant de représenter les différents éléments de l'application :

- Une classe Connectome qui sert à stocker la matrice d'adjacence et l'atlas d'un connectome
- Une classe Graph qui sert à stocker le graphe construit à partir du connectome
- Une classe AntAlgorithm qui sert à mettre en place l'algorithme de colonie de fourmis

Après révision, la partie Modèle comprend les différentes classes permettant de représenter les éléments permettant de gérer les connectomes :

- Une classe ConnectomeObject qui permet de regrouper les éléments permettant de définir et gérer les connectomes
- Une classe ConnectomeGraph qui va permettre de modéliser le connectome sous forme de graph NetworkX
- Une classe BrainVolume qui va permettre de modéliser le cerveau sur lequel est basé le connectome

## 2.2 Vue

De la même manière que le modèle, la partie vue a évolué après la révision du diagramme de classe :

La Vue quand à elle comportera 3 éléments :

- La classe MainWindow qui représente la fenêtre principale de l'application où se trouve les boutons pour effectuer les différentes actions et la navigation
- La classe GraphVisualizer pour afficher à l'écran une visualisation du graphe sélectionné
- La classe ComparisonViewer qui permet d'afficher les résultats de la comparaison entre les graphes

Après révision, les classes de la vue sont :

- Une classe MainWindow qui représente la fenêtre principale de l'application où se trouve les boutons pour effectuer les différentes actions et la navigation
- Une classe ImportWindow qui affiche un formulaire pour la création d'un nouvel objet connectome
- Une classe CompareWindow qui affiche la comparaison de plusieurs graphes de connectomes
- Une classe VisualizeWindow qui affiche la visualisation 3D d'un graphe de connectome

En pratique, la visualisation se fait dans la classe ConnectomeVisualizer qui va directement visualiser le connectome après création.

## 2.3 Contrôleur

Le Contrôleur comporte les classes qui effectueront les traitements sur les connectomes et les graphes :

- La classe FileManager qui va gérer l'import des données d'entrée et la sauvegarde des différents éléments
- La classe GraphMaker qui va gérer la transformation d'un connectome en graphe compatible avec l'algorithme de comparaison

- La classe GraphComparer qui va appliquer l'Ant Colony Optimization pour comparer les graphes générés

Après révision du diagramme de classe, les contrôleur sont les classes permettant de gérer les différentes fonctions des fenêtres de la partie vue.

## 2.4 Classes à la fin du projet

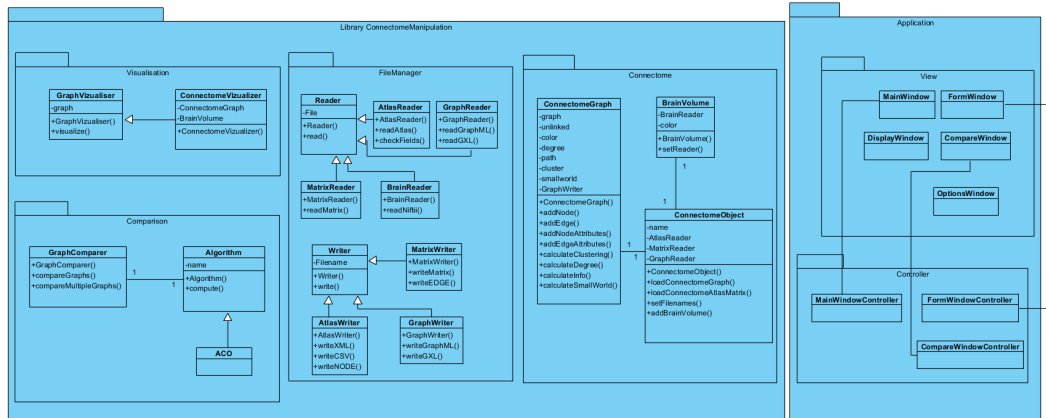


Figure 3 – Diagramme de classe final

On peut voir qu'à la fin du projet, le diagramme de classe est différent du diagramme de classe prévu au début de la phase de développement. En effet, lors de l'ajout des différentes fonctionnalités qui n'étaient pas prévues lors de la réalisation du premier diagramme, de nouvelles classes ont été créées pour correspondre aux classes ajoutées.

La principale différence est dans les classes de l'application, puisque plus de fenêtres ont été créées que prévu. Ce diagramme de classe ne présente aussi pas de manière exhaustive les fonctions des différentes classes et ne montrent que les fonctions principales prévues au début du développement, puisque les fonctions intermédiaires qui font appel à d'autres fonctions ne sont pas pertinentes dans le diagramme de classe. Pour la même raison, les classes de Contrôleurs et Vues ne montrent pas leurs fonctions et attributs.

# 5

## Mise en œuvre

Dans cette partie, on va parler de la réalisation du projet et de la partie développement de l'application qui a été effectuée au semestre 10.

### 1 Outils et librairie utilisées

Au cours de ce projet, il est question de développer une application permettant de gérer des connectomes sous forme de graphe. Certaines limitations ont été imposées notamment par rapport au langage de programmation, et aux formats de graphe utilisés.

Ainsi, le projet est développé en Python avec l'utilisation de plusieurs librairies utilisées pour les différents modules :

- Une des librairies les plus utilisée dans ce projet est la librairie NetworkX qui permet la gestion de graphe en Python. Cette librairie va nous permettre de construire des graphes composés de sommets et d'arêtes à partir des fichiers de matrice d'adjacence et d'atlas des connectomes. Elle permet aussi de gérer certains formats de fichiers de graphe pour pouvoir lire et écrire des graphes de connectomes.
- Pour la lecture des fichiers d'initialisation (matrice et atlas), il existe plusieurs formats possibles de lecture. On a donc pour certains formats de fichiers utilisé les librairies adéquates, notamment la librairie csv pour les fichiers csv et la librairie xml pour les fichiers xml.
- L'application nécessite une interface graphique, et la priorité de ce projet étant plus axée sur la fonctionnalité que l'esthétique, on a décidé d'utiliser la bibliothèque Tkinter pour réaliser une interface graphique légère qui se concentre sur l'efficacité.
- Une des fonctionnalités nécessaire est la visualisation des graphes en 3D, que l'on fait avec la bibliothèque plotly pour visualiser des réseaux dans un environnement 3D.
- La visualisation fonctionne avec du code HTML, et pour éviter le besoin d'une connexion internet, on utilise la bibliothèque webview pour simuler un navigateur web dans une fenêtre.

Pour avoir une trace des tâches à faire et faites, un planning Trello a été utilisé.

Un dépôt github a été créé pour le versioning et la distribution du code.

## 2 Implémentation

Le développement du projet s'effectue en module, c'est-à-dire que le développement d'un module concerne une fonctionnalité de l'application, et que l'on passe ensuite à la fonctionnalité suivante une fois une fonctionnalité terminée.

### 2.1 Limites et problèmes rencontrés

Une des principales problématiques du projet est le test des fonctionnalités. En effet, les connectomes représentant des graphes massifs et ne possédant pas de véritables terrains, il est difficile d'effectuer des tests sur les graphes générés à partir des fichiers matrice et atlas. On se rabat donc sur des tests des différentes fonctionnalités en supposant que la lecture des fichiers s'effectue correctement et que le graphe généré correspond.

Un autre problème s'est assez rapidement posé. Le premier module développé concerne la lecture des fichiers et la génération de graphes à partir de ces fichiers. Ces fichiers sont construits en amont de l'utilisation de l'application par les utilisateurs et peuvent être de plusieurs types de fichiers (xml, csv, txt, edge...). De plus, même au sein d'un même type de fichier, la disposition des attributs et la syntaxe peut être différente.

Après avoir étudié les possibles solutions, il était apparent qu'il est impossible de gérer tout les éventuels types de fichiers et format, on a donc décidé de construire un lecteur de fichier qui va identifier la syntaxe et générer au mieux le graphe correspondant, tout en restreignant les types de fichiers lisibles par l'application.

Après avoir développé le premier module, une discussion a eu lieu pour revoir le développement de l'application et produire un rendu différent : au lieu de ne développer que du code propre à l'application, il était plus intéressant de développer une librairie de manipulation de connectomes plutôt que de faire du code qui ne sera pas réutilisable.

Évidemment, cela inclut une charge de travail plus grande et une période de recherche pour estimer la difficulté et la charge de travail pour rajouter ces changements. Ainsi, il a fallu faire une refonte du planning et du diagramme de classe, et la décision a été prise de se concentrer principalement sur la qualité plutôt que la quantité pour produire des modules fonctionnels même si cela implique de ne pas finir le projet.

Certaines fonctionnalités ont donc été reléguées à un rang plus bas, et seront donc omises ou développées en fin de projet si le temps le permet. La visualisation va par exemple se faire dans un premier temps sans image de cerveau, la visualisation est indisponible si l'atlas ne possède pas de coordonnées 3D, et seuls certains types de graphes sont disponibles en écriture et lecture (dans un premier temps graphml et gxl si le temps le permet).

### 2.2 Choix d'implémentation

Certaines parties du projet n'avaient pas d'indication spécifique sur l'implémentation autre que d'utiliser le langage Python et le résultat attendu. Des choix ont donc été faits pour la réalisation de certaines fonctions prenant en compte différents paramètres qui seront décrits ci-dessous.

Pour la partie visualisation, la visualisation devait permettre une représentation 3D des graphes, donc je me suis tourné dans un premier temps vers matplotlib, mais les capacités de matplotlib ne m'ont pas paru suffisamment puissantes pour les graphes de connectomes, donc le choix s'est porté sur Plotly. Cependant, Plotly est une solution web, donc pour pouvoir utiliser Plotly de manière intégrée dans l'application, il a été nécessaire d'utiliser la bibliothèque Pywebview pour

pouvoir ouvrir une fenêtre Tkinter qui utilise le code HTML de Plotly pour pouvoir l'afficher de manière offline et dans l'application. Cela permet de profiter des capacités de Plotly tout en intégrant l'affichage à Tkinter.

Cependant, cette implémentation ne prête pas attention à des problématiques d'efficacité, de mémoire ou de performance, et l'implémentation a été faite avec un objectif de fonctionnalité pur.

Pour la partie création de graphe de connectomes, le lecteur d'atlas doit pouvoir lire plusieurs types de fichiers. Un des choix d'implémentation qui a été fait est la création d'une partie "polyvalente" du lecteur qui va tenter de reconnaître les champs de l'atlas s'ils ne font pas partie des formats classiques. En effet, il est facile de faire des lecteurs pour les fichiers d'atlas qui ont une syntaxe spécifiques, comme les fichiers d'atlas de Brain Net Viewer et FSL, mais des atlas peuvent provenir d'autre sources, comme par exemple l'Hôpital de Tours, et ne pas suivre les conventions d'atlas classique. Pour cela, on a décidé de faire un lecteur capable d'identifier les types de variables dans les champs et d'attribuer ces valeurs à des valeurs attendues pour un atlas (index du sommet, nom du sommet, coordonnées...). Mais un atlas peut porter des informations plus spécifiques, comme l'hémisphère du sommet où le degré du sommet, qui ne peuvent pas être reconnues par ce lecteur.

Le choix a donc été de faire un lecteur qui permet de lire de manière imparfaite mais qui permet de lire des fichiers autres que les formats qui ont des syntaxes prédéfinies plutôt que de se limiter à ces formats.



# 6

## Conclusion

### 1 Bilan du semestre 9

Au cours de ce semestre, les tâches effectuées étaient plus du domaine de la recherche que du développement. En effet, la majorité du temps de travail de cette première partie du PRD a été consacrée à la construction d'une littérature pour l'état de l'art et la recherche de différentes technologies pour la réalisation du projet. Cependant, un premier travail préparatoire a été effectué pour la deuxième phase du projet au cours du S10.

#### 1.1 Tâches effectuées au S9

- Compréhension du sujet
- Étude de la faisabilité du projet
- Identification des différents modules à développer
- Recherches sur les outils nécessaires au projet
- Rédaction du cahier des spécifications
- Mise en place d'une gestion de projet
- Rédaction de la partie S9 du rapport
- Préparation d'un jeu de donnée et d'un plan de tests

#### 1.2 Tâches en retard au S9

Lors de cette première partie du projet de Recherche et Développement, il n'y a pas eu de répartition des tâches ordonnées à l'aide d'outils tels que le diagramme de Gantt.

Étant donné la nature volatile de la phase de recherche, le déroulement de la première phase ne suivait pas de plan particulier.

A la fin de ce premier semestre de recherche, un retard a été pris quant à la définition du sujet. Ce retard a été entraîné par une mauvaise compréhension de l'exercice de rédaction d'un cahier de spécification, et donc une mauvaise définition du sujet et du travail à réaliser. Du travail supplémentaire a donc été nécessaire pour corriger ces erreurs.

Au moment de la finalisation du S9, les tâches sont à jour en concordance avec les livrables demandés.

### 1.3 A faire au S10

Le S10 sera principalement concentré sur le développement de l'application. Un certain nombre de tâches de développement seront donc à réaliser :

- Développement d'une interface graphique
- Développement d'un module de génération de graphes
- Développement d'un module de visualisation
- Développement d'un module de comparaison
- Mise en place d'une procédure de test pour les modules de l'application
- Déploiement d'un outil de versioning
- Mise en place d'outils de gestion de projet (assurance qualité, planning)
- Rédaction de la seconde partie du rapport

Le déroulement de cette seconde partie sera en méthode agile avec des sprints d'une à deux semaines et des réunions hebdomadaires ou bi-mensuelles si possible.

## 2 Bilan du semestre 10

Ce second semestre est principalement consacré au développement du projet, et à la rédaction des différents documents qui permettent la continuation du projet. Globalement, le second semestre s'est mieux déroulé que le premier puisque le développement est une compétence plus maîtrisée que la rédaction du cahier des spécifications. Cependant, des répercussions des erreurs du premier semestre se sont faites sentir au cours du second.

### 2.1 Tâches effectuées au S10

Au cours du second semestre, les tâches à faire étaient principalement le développement des modules spécifiés au premier semestre. A cela s'ajoute des tâches de rédaction de certains documents et de l'étude du projet et de la continuation du projet.

Les tâches réalisées au cours de ce semestre sont donc :

- Redéfinition du cadre du projet pour inclure une bibliothèque de manipulation de connexions
- Développement d'une interface graphique
- Développement d'un module de génération de graphes
- Développement d'un module de visualisation
- Mise en place d'une procédure de test pour les modules de l'application
- Déploiement d'un outil de versioning
- Mise en place d'outils de gestion de projet (assurance qualité, planning)
- Rédaction de la seconde partie du rapport
- Rédaction de documents de continuation ( cahier de développeur, guide d'installation et d'utilisation)
- Développement de fonctionnalités non spécifiées dans le cahier des spécifications

Comme on peut le voir, cette liste de tâches réalisées est différente de la liste de tâches prévues pour le S10 au S9. Cette différence provient du fait qu'il y a une redéfinition du cadre du projet et on a donc changé la priorité de certaines fonctionnalités et dévié du cahier de spécification du premier semestre.

## 2.2 Reste à faire et retards

Le cadre du projet ayant été élargi, l'ampleur du projet a dépassé la quantité de travail du PRD et donc certaines fonctionnalités ont dû être abandonnées pour laisser places à d'autres tâches jugées plus urgentes. Il reste donc à faire certaines fonctionnalités du cahier des spécifications ainsi que d'autres fonctionnalités mentionnées au cours de cette redéfinition du projet.

Parmi ces fonctionnalités, on trouve :

- Le module de comparaison qui a été commencé à titre d'exemple, mais qui ne possède pas d'implémentation d'algorithme de comparaison comme l'ACO
- Certaines fonctionnalités de lecture et d'écriture pour supporter plus de formats de fichier graphe ou de matrice d'adjacence/atlas. Le format de graphe gxl serait à ajouter en lecture et écriture
- Certaines fonctionnalités de visualisation pour inclure un modèle 3D de cerveau autour du graphe pour visualiser la position des points dans le cerveau
- Correction de certains bugs/défaut de l'interface graphique et amélioration globale de l'interface

Là où le module de comparaison et les formats de fichiers supplémentaires ont été relégués en tant que tâches moins importantes et laissées à faire, les autres fonctionnalités mentionnées n'ont pas été terminées par manque de temps où pour laisser place à d'autres fonctionnalités et sont donc incomplètes où imparfaites et possèdent tout de même une première base de code qui pourra être repris par la suite.

Dans l'ensemble, étant donné la révision du planning et le rythme de travail agile qui poussait à ajouter de nouvelles fonctionnalités de manière régulière, le projet a tout de même progressé à un rythme raisonnable et la plupart des demandes du client ont été atteintes.

## 2.3 Bilan sur la qualité

Le projet s'est, dans son ensemble, bien déroulé, et la phase de développement a progressé sans problème majeur et la majorité des fonctionnalités ont été développées. Plusieurs documents ont été produits pour documenter de l'assurance qualité du projet :

- Un cahier de test qui documente les tests effectuer pour vérifier le bon fonctionnement des fonctionnalités de l'application
- Un cahier de développeur qui détaille le projet et l'environnement de développement pour permettre une bonne reprise du projet
- Un guide d'utilisation de l'application pour explorer les différentes fonctionnalités
- Un guide d'installation de l'application pour pouvoir lancer l'application

Ces documents et la complétion des fonctionnalités demandées témoignent d'une mise en œuvre qualité pour ce projet.

Cependant, de nombreuses fonctionnalités ajoutées ne sont pas spécifiées dans le cahier des spécifications, il est donc difficile de faire le bilan sur la qualité du projet par rapport à ce qui est présent dans le cahier des spécifications et la première partie du rapport, donc les spécifications fonctionnelles ne sont plus cohérentes avec les livrables délivrés à la fin du projet.

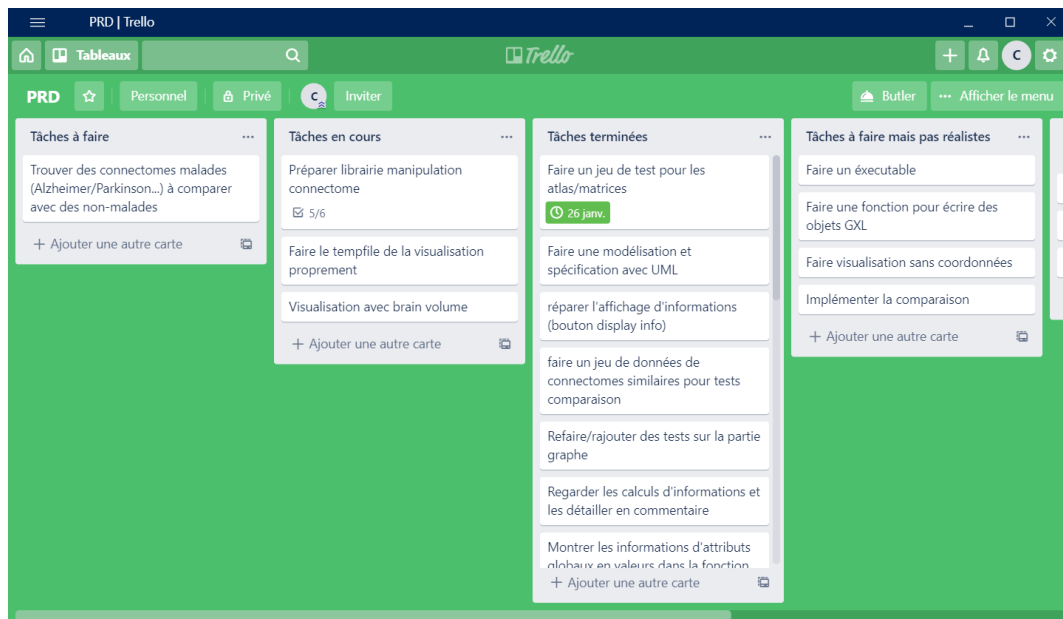


Figure 1 – Trello des tâches de la phase de développement

Un outil Trello a été mis en place pour suivre les différentes tâches de cette seconde partie et leurs avancées, ce qui permet de suivre les fonctionnalités ajoutées ainsi que les différentes problématiques rencontrées qui ne font pas partie de la phase de développement.

## 2.4 Bilan sur la gestion du projet

Ce projet était le premier projet informatique de cet envergure sur lequel j'ai travaillé, et un véritable effort d'organisation et de gestion de projet a été demandé. J'ai dû au cours de ce projet travailler des compétences qui ne sont pas ou peu maîtrisées et j'ai donc rencontrés des difficultés sur certains aspects du projet.

Globalement, beaucoup moins de problèmes ont été présents sur la deuxième partie du projet, c'est-à-dire la phase de développement. En effet, mes compétences de développement sont beaucoup plus avancées que mes compétences de gestion de projet, et j'ai donc eu plus de facilité lors de cette seconde partie.

Cependant, il y a tout de même eu des difficultés au cours de cette seconde partie, et des éléments qui auraient pu être améliorés.

Le premier point sur lequel j'aurais pu corriger mon projet est de mieux profiter du fait que j'avais 3 encadrants sur ce projet. J'ai en effet eu tendance à ne travailler quasiment exclusivement avec Diem, avec qui je faisais la plupart de mes rencontres pour discuter du projet et avec laquelle je prenais les suggestions de fonctionnalités pour le projet. J'aurais pu, et dû, au cours de cette phase de développement, aussi interagir avec mes deux autres encadrants pour d'autres aspects du projet (comme les aspects qualités et gestion de projet) qui font moins partie du domaine d'expertise de Diem. Une piste d'amélioration de ma gestion de projet aurait donc été de plus profiter de mes encadrants.

Certains aspects de la gestion de projet n'ont aussi pas été assez poussés. Certains éléments étaient tout simplement manquants. J'en ai par exemple pas fait d'analyse de risque pour le projet, ce qui aurait pu me servir à mieux organiser mon travail ce second semestre.

Pour résumer, les lacunes du premier semestre, notamment sur la partie définition du cadre du projet et spécifications ont définitivement impacté négativement la phase de développement, mais le travail a tout de même été réalisé à un bon rythme et la plupart des demandes ont été atteintes. Je considère donc ce second semestre est comme une réussite.

**Deuxième partie**

**Documents annexes**

# 7

## Description des interfaces externes du logiciel

### 1 Interfaces matériel/logiciel

Le matériel nécessaire à l'utilisation de cette application se limitera à un équipement de bureau classique ainsi que Python installé sur l'ordinateur. Une connexion internet sera aussi requise en fonction de la solution retenue. Une connexion internet permettra de télécharger les informations d'atlas classiques si un atlas n'est pas fourni.

### 2 Interfaces homme/machine

Le logiciel se présentera sous la forme d'une application simple permettant d'effectuer toutes les opérations depuis une même fenêtre :

- Une fenêtre principale possédant un menu et des boutons permettra d'effectuer l'import et la sélection des fichiers à utiliser ainsi que de choisir les opérations à effectuer.
- La visualisation se fera via un onglet/fenêtre ou via un navigateur en fonction de la solution choisie par l'utilisateur.
- Les graphes générés pourront être exportés ou traités et comparés directement via l'application. Les graphes pourront être exportés au format GraphML ou GXL. Ci-dessous sont présentés des mockup de différentes fenêtres de l'application.

### 3 Interfaces logiciel/logiciel

Des bibliothèques Python seront utilisées pour effectuer des traitements de neuroimagerie. On retrouvera parmi ces bibliothèques Nilearn, Nibabel et Nipype.

La bibliothèque Python Nipype permet l'interfaçage avec de nombreuses applications de neuroimagerie capables de manipuler les données, visualiser des connectomes ou effectuer des traitements.

On utilisera la bibliothèque NetworkX pour générer les graphes à partir d'informations de connectomes.

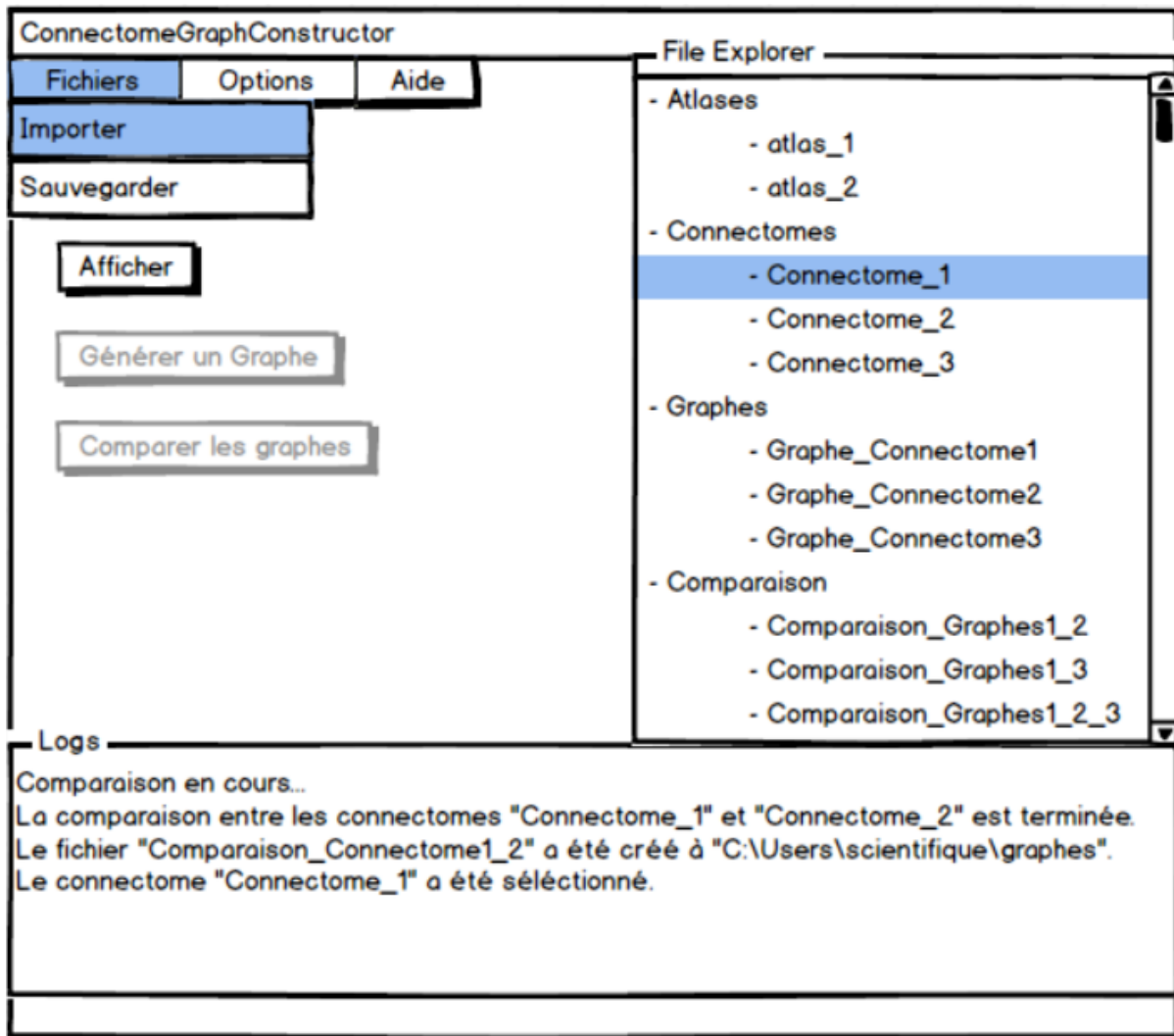


Figure 1 – Mock-up de la fenêtre principale de l'application

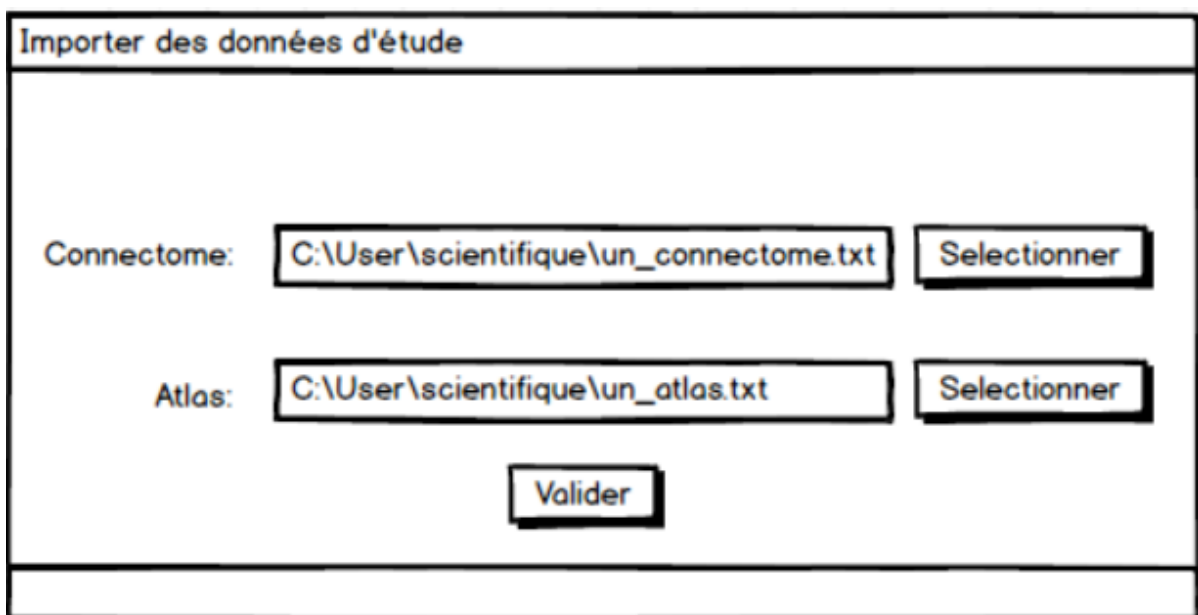


Figure 2 – Mock-up de la fenêtre d'import de données

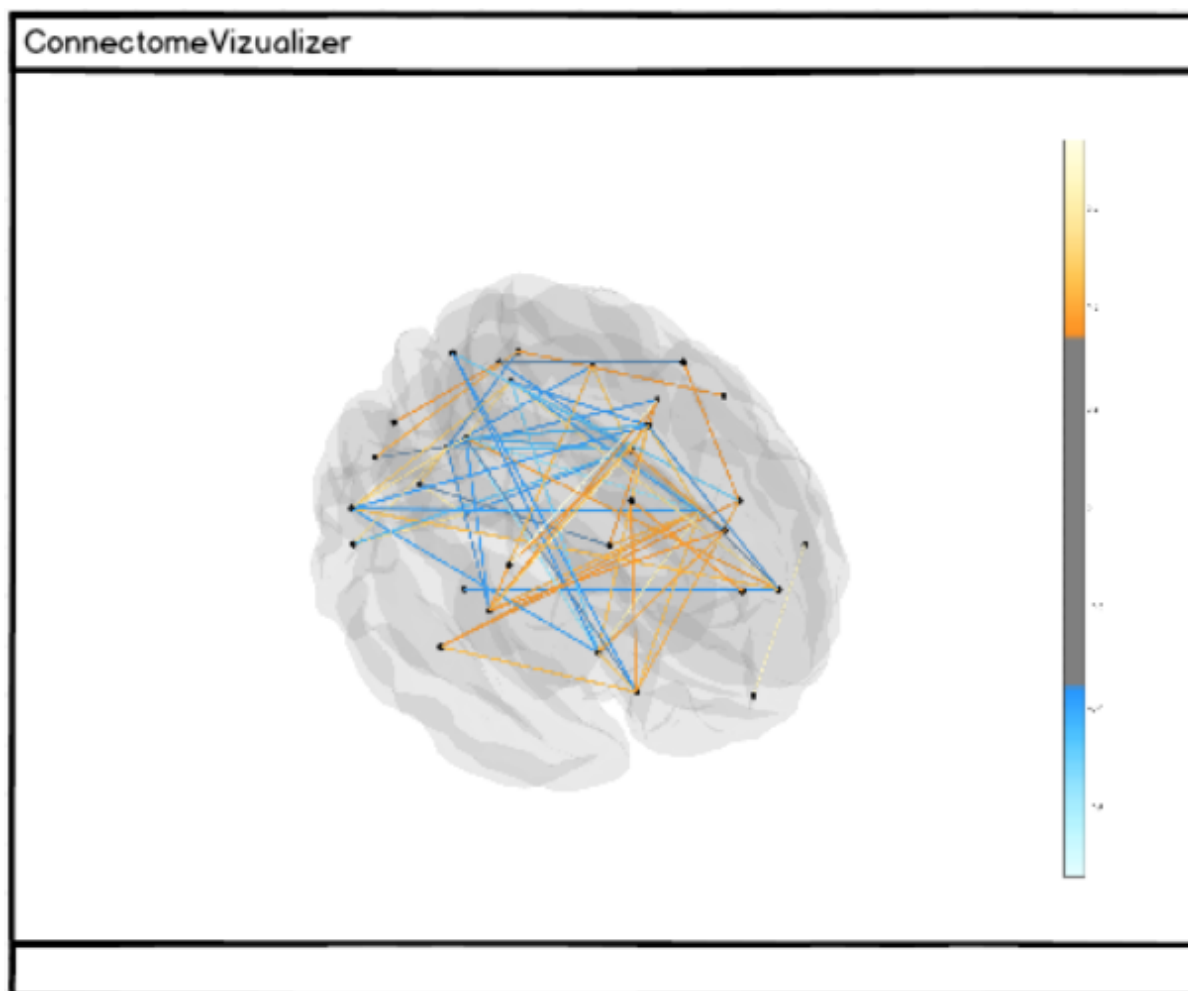


Figure 3 – Mock-up de la fenêtre de visualisation de connectomes



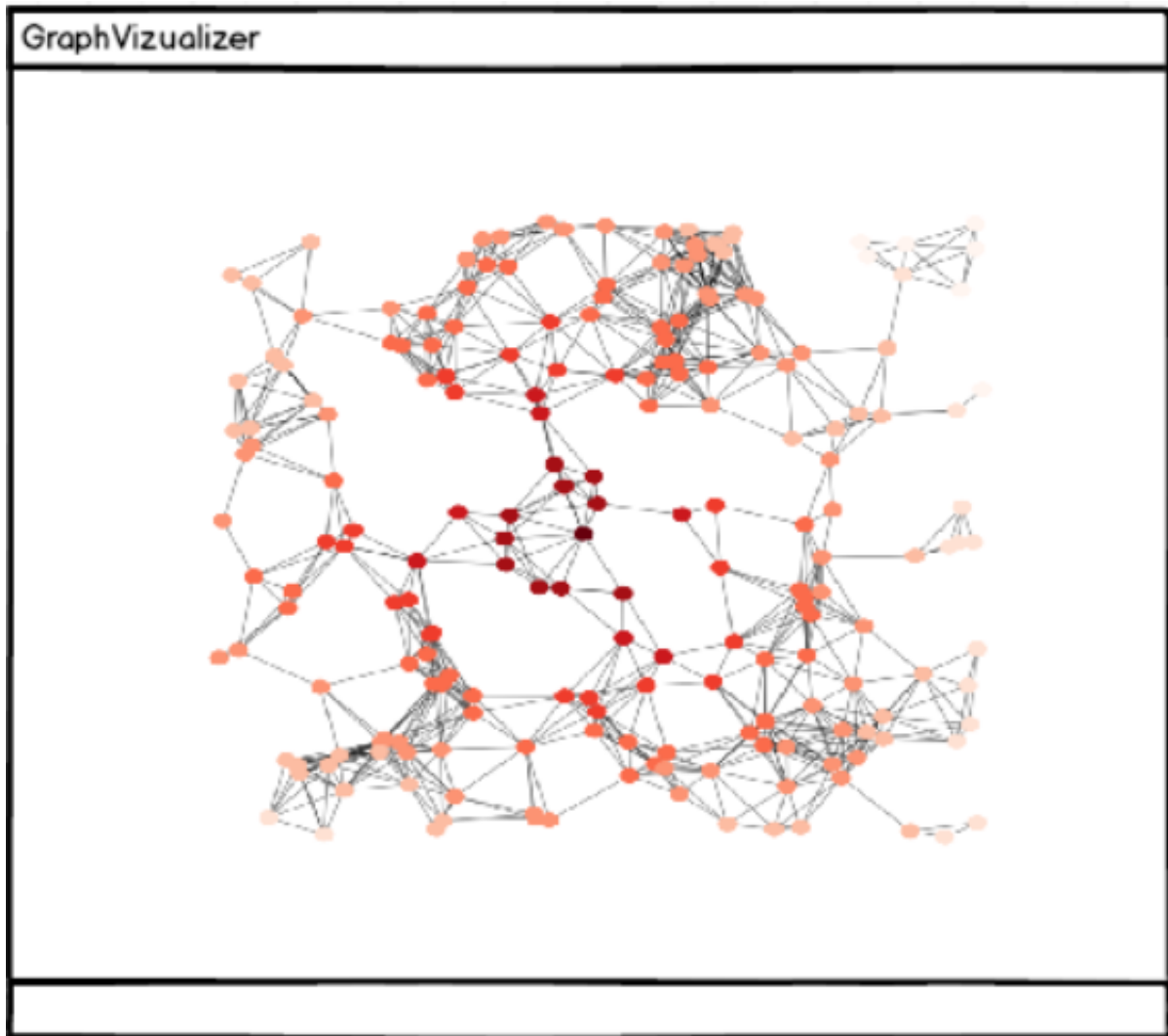


Figure 4 – Mock-up de la fenêtre de visualisation de graphe

# 8

## Spécifications fonctionnelles

### 1 Définition des fonctions

#### 1.1 Définition de la fonction 1

Identification de la fonction 1 : importConnectomeData

Fonction permettant d'importer les données concernant un connectome dans l'application.

**Priorité** : primordiale

Description de la fonction 1

**Entrée :**

- Un fichier texte contenant la matrice d'adjacence du connectome.
- Un fichier texte contenant l'atlas utilisé pour construire le connectome.
- Une chaîne de caractère représentant le titre du connectome.

**Sortie :**

- Un booléen témoignant de la réussite ou l'échec de l'import.

Cette fonction va stocker l'atlas et la matrice d'adjacence dans des variables pour pouvoir construire un graphe.

L'utilisateur va pouvoir lors de l'import définir un nom pour le connectome qui servira pour les affichages et la construction du graphe. Les fichiers d'entrées doivent contenir à minima les données pour construire un graphe binaire et non orienté, avec des labels pour les sommets. L'utilisateur pourra à l'import choisir un titre pour le graphe modélisé à partir de ce connectome.

#### 1.2 Définition de la fonction 2

Identification de la fonction 2 : saveGraph

Fonction permettant de sauvegarder le graphe sélectionné dans un fichier.

**Priorité** : faible

Description de la fonction 2

**Entrée :**

- Le graphe sélectionné par l'utilisateur.

**Sortie :**

- Un booléen témoignant de la réussite ou l'échec de l'enregistrement.

Cette fonction va permettre l'enregistrement du graphe dans un fichier.

Une fenêtre de dialogue permettra à l'utilisateur d'entrer un nom pour enregistrer le fichier et choisir le format. Le graphe sera enregistrable au format GXL(Graph eXchange Language) ou GraphML.

Dans le cas où l'enregistrement est impossible, une description de l'erreur apparaîtra en rouge dans les logs, décrivant le problème.

**1.3 Définition de la fonction 3**Identification de la fonction 3 : connectomeToGraph

Fonction permettant de transformer un ou plusieurs connectome en un graphe compatible avec les algorithmes de comparaison.

**Priorité :** primordiale

Description de la fonction 3**Entrée :**

- Un tableau contenant les matrices d'adjacence correspondant aux connectomes.
- La variable contenant les informations de l'atlas.

**Sortie :**

- Un graphe représentant le connectome construits avec les données fournies en entrée.

Cette fonction construit un graphe NetworkX à partir d'un ou plusieurs connectomes et des informations présentes dans l'atlas. Chaque arête possèdera les informations de tous les connectomes correspondant à cette arête, et les sommets seront les différents sommets de l'atlas.

Comme indiqué dans la section Hypothèses, si le temps le permet, on ajoutera à cette fonctionnalité la possibilité pour l'utilisateur de sélectionner des options supplémentaires d'information sur le graphe.

Pour qu'un graphe soit construit avec des données de plusieurs connectomes, il faut que tous les connectomes entrant soit basés sur le même atlas.

**1.4 Définition de la fonction 4**Identification de la fonction 4 : visualizeBrainConnectome

Fonction permettant d'afficher un connectome sous forme de graphe.

**Priorité :** primordiale

Description de la fonction 4**Entrée :**

- La matrice d'adjacence du connectome.
- La variable contenant les informations de l'atlas.
- Le seuil de précision du connectome(optionnel).

**Sortie :**

- Pas de sortie.

Cette fonction affiche une visualisation minimale du connectome sous forme de graphe.

Dans le cas où l'utilisateur choisit la version interactive, un fichier html temporaire sera généré et ouvert dans le navigateur par défaut. L'utilisateur pourra effectuer des rotations pour observer le connectome sous différents angles, zoomer ou capturer une image au format (.png).

Le seuil de précision indique le nombre de connexions que l'on veut afficher. L'utilisateur peut

ainsi décider d'afficher uniquement les 20% connexions les plus fortes pour plus de visibilité par exemple.

## 1.5 Définition de la fonction 5

Identification de la fonction 5 : visualizeGraph

Fonction permettant d'afficher le graphe contenu dans le fichier en entrée.

**Priorité** : primordiale

Description de la fonction 5

**Entrée** :

- Un graphe NetworkX à afficher au format graphML ou GXL.
- Un entier représentant l'option d'affichage du graphe (2D ou 3D).

**Sortie** :

- Pas de sortie.

Cette fonction affiche le graphe sélectionné. Le graphe pourra être construit en 2D ou en 3D. Si les sommets ne possèdent pas de coordonnées 3D et que l'affichage sélectionné est 3D, le graphe sera généré dans un environnement 3D avec des coordonnées aléatoires pour les sommets.

## 1.6 Définition de la fonction 6

Identification de la fonction 6 : compareGraphs

Fonction permettant de comparer deux graphes ou plus.

**Priorité** : primordiale

Description de la fonction 6

**Entrée** :

- Un tableau de graphes construits avec NetworkX représentant des connectomes.
- Les paramètres de configuration de l'ACO.
- Un emplacement pour créer le fichier de résultat.

**Sortie** :

- Un fichier décrivant le résultat de la comparaison entre les graphes.

La comparaison sera faite à l'aide de l'[Ant Colony Optimization](#) sur le multivalent graph matching qui permettra de mettre en valeur les différences entre les deux graphes.

Dans le cas où un seul graphe est fourni, la comparaison sera effectuée avec un graphe de référence.

Un affichage du résultat de la comparaison sera aussi fait dans les logs de l'interface graphique.

Un fichier contenant le résultat de la comparaison sera généré.

# 9

## Spécifications non fonctionnelles

### 1 Contraintes de développement et conception

L'application aura besoin pour fonctionner d'un environnement Python. De plus il faudra que l'environnement Python soit équipé des librairies nécessaires au lancement de l'application. Une connexion internet sera requises pour la visualisation interactive des connectomes. On va dans un premier temps viser les machines Windows, puis éventuellement d'autres OS si le temps est disponible.

Le développement sera en Python à l'aide de bibliothèques telles que Nipype, Nibabel et Nilearn qui permet le traitement de neuroimagerie et l'interfaçage avec d'autres applications de neuroimagerie. La bibliothèque NetworkX sera utilisé pour la construction de graphes.

### 2 Contraintes de fonctionnement et d'exploitation

#### 2.1 Performances

On souhaite que l'application soit réactive, idéalement toutes les opérations de navigation/sélection dans le logiciel devraient individuellement s'effectuer en moins d'une seconde.

Pour les opérations de modélisation, c'est-à-dire l'affichage du graphe ou du connectome, le temps dépendra de la complexité du graphe à afficher. Si l'utilisateur sélectionne des informations supplémentaires à afficher sur les arêtes et sommets, le temps d'exécution sera augmenté. La comparaison avec l'algorithme de colonie de fourmis est aussi une opération coûteuse, la durée dépendra des paramètres choisis.

La vitesse des opérations de calculs, notamment lors de la comparaison et de la visualisation dépendront de la puissance de la machine.

#### 2.2 Capacités

Les données d'entrées sont des fichiers représentant des matrices d'adjacence et des informations d'atlas, la taille des fichiers restera donc réduite.

Dans le cadre où des fichiers de neuroimagerie seront stockés pour des opérations de préparation, on a cependant des fichiers d'IRM de plusieurs Go.

### 2.3 Contrôlabilité

Une fenêtre sera disponible pour consulter les logs et s'assurer du bon fonctionnement des opérations.

L'utilisateur pourra paramétrer les opérations pour contrôler le temps d'exécution, mettre un timeout ou la précision du connectome généré.

Une possibilité de gérer plus d'options sera envisageable si le temps le permet.

### 2.4 Sécurité

L'application ne traitant pas de données sensibles, la question de sécurité ne s'applique pas.

### 2.5 Intégrité

L'application va devoir créer des données au cours de son utilisation, et un dysfonctionnement pourrait engendrer la perte de données. Les données manipulées n'étant pas critiques et l'application n'agissant sur les données d'entrée qu'en lecture seule, lors d'une déconnexion imprévue l'application abandonnera l'opération en cours qui devra être recommencée.

Lorsqu'un problème se produit, les opérations sont annulées et si l'opération inclut la création d'un fichier, aucun fichier n'est créé.

# 10

## Livrables et conduite de test

### 1 Gestion des livrables au cours de la phase de développement

Le développement de l'application s'effectue en développement de différents modules, un livrable sera donc fourni après chaque phase de test des modules. Les modules seront d'abord livrés indépendamment avec le jeu de test utilisé pour la validation du fonctionnement puis un livrable réunissant les modules avec l'interface graphique sera livré.

Le premier livrable avec interface graphique sera livré lorsque tous les modules gérant les connectomes seront fonctionnels avec une première version de l'interface graphique.

Le module de comparaison de graphe sera ensuite fourni après avoir été testé.

Enfin après une phase de tests finale, un livrable contenant tous les modules ainsi qu'une interface graphique finalisé sera livré.

Le client étant disponible sur place et des rencontres régulières étant possibles, on va adopter une méthodologie agile de type SCRUM pour la phase de développement. On propose tout de même une estimation du plan de développement des modules à l'aide du diagramme de Gantt suivant :

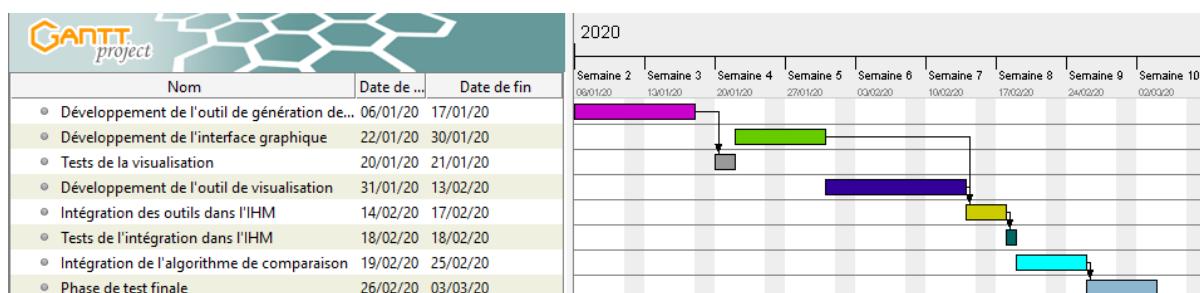


Figure 1 – Diagramme de Gantt de la phase de développement

Les modules seront d'abord livrés indépendamment avec le jeu de test utilisé pour la validation du fonctionnement puis un livrable réunissant les modules avec l'interface graphique sera livré.

Le premier livrable avec interface graphique sera livré lorsque tous les modules gérant les connectomes seront fonctionnels avec une première version de l'interface graphique.

Le module de comparaison de graphe sera ensuite fourni après avoir été testé.

Enfin après une phase de tests finale, un livrable contenant tous les modules ainsi qu'une interface graphique finale sera livré.

## 2 Plan de conduite de test

L'application fonctionne à partir de données de connectomes construits à partir d'imagerie médicale (d'IRM fonctionnelle ou de diffusion en fonction du type de connectomes représentés). Il faudrait donc effectuer des jeux de tests provenant d'au moins un connectome fonctionnel et un connectome structurel.

Les tests suivants seront effectués sur des données générées par l'application. Ne possédant pas de connectome de référence ou d'expert capable d'assurer si les connectomes générés sont pertinents, les tests seront consacrés à l'aspect logiciel de l'application.

De plus, les données volumineuses telles que celles du HCP nécessitant une puissance élevée, les tests seront faits sur des échantillons moins complexes.

Les modules de visualisation et de génération de graphe seront testés sur un connectome construit à partir des données suivantes, et sur le graphe généré à partir de ce connectome :

- L'image `NiFTI filtered_func.nii` obtenable sur le site du format de `fichier NiFTI` qui représente des séries temporelles d'un connectome fonctionnel
- L'atlas `MSDLatlas` par G.Varoquaux, A ; Gramfort, F. Pedregosa, V. Michel et B. Thirion

Ces données permettront la construction d'un connectome fonctionnel simple sur lequel on pourra tester les modules. Ce connectome sera construit avant la phase de test à l'aide des bibliothèques Nilearn et Nibabel et les données de l'atlas et de la matrice d'adjacence seront enregistrées dans des fichiers textes.

Pour le connectome structurel, on va utiliser un connectome construit à partir des données de l'hôpital de tours. Le connectome a été construit à partir de l'image `NiFTI aparc.a2009s+aseg.nii` et des fichiers ont été créés contenant les informations du connectome :

- Le fichier `liste_regions_valides.txt` contient la répartition en régions qui permet de construire les sommets
- Le fichier `MatConnEnds.txt` contient la matrice d'adjacence des connexions entre régions finales
- Le fichier `MattConn.txt` contient la matrice d'adjacence complète

La différence entre les fichiers `MattConn` est la suivante : pour une connexion allant de la région A à la région C en passant par la région intermédiaire B, le fichier `MattConnEnds` ne contient qu'une connexion A vers C là où `MattConn` contient les connexions A vers B et B vers C. On utilisera dans le cadre de nos tests un connectome construit avec la matrice d'adjacence contenue dans `MattConn.txt`.

Le module de comparaison de graphes sera testé en effectuant des comparaisons de deux graphes différents, deux graphes identiques puis une comparaison de 3 graphes. Le résultat de la comparaison sera ensuite testé à partir du fichier généré.

## 3 Condition de validation de test

Test du module de visualisation :

**Validation :** La sélection d'un connectome affiche correctement une représentation graphique du connectome construit à partir de la matrice d'adjacence et de l'atlas. La sélection d'un graphe au format graphML ou GXL affiche une modélisation 2D ou 3D du graphe sélectionné.



Test du module de génération de graphe :

**Validation** : Un graphe NetworkX possédant des sommets et arêtes correspondant au connectome d'entrée est généré.

Les sommets et arêtes du connectome doivent tous être présents dans le graphe, et le poids des arêtes doit correspondre à l'intensité des connexions si la matrice d'adjacence donnée n'est pas binaire.

Si plusieurs connectome sont donnés en entrée, un graphe unique est généré et chaque arête et sommets contient les informations propres à chaque connectome d'entrée.

Test du module de comparaison de graphes :

**Validation** : Le résultat de la comparaison correspond aux différences entre les graphes de connectomes sélectionnés. La comparaison de deux graphes identiques ne doit retourner aucune différence et la comparaison de trois graphes doit retourner les différences individuelles entre chaque graphe.

Les tests seront effectués sur les jeux de données décrits plus hauts. En raison de leur petite taille, il est facile de décerner la présence d'une erreur.

## 1 Diagramme de classe et explications

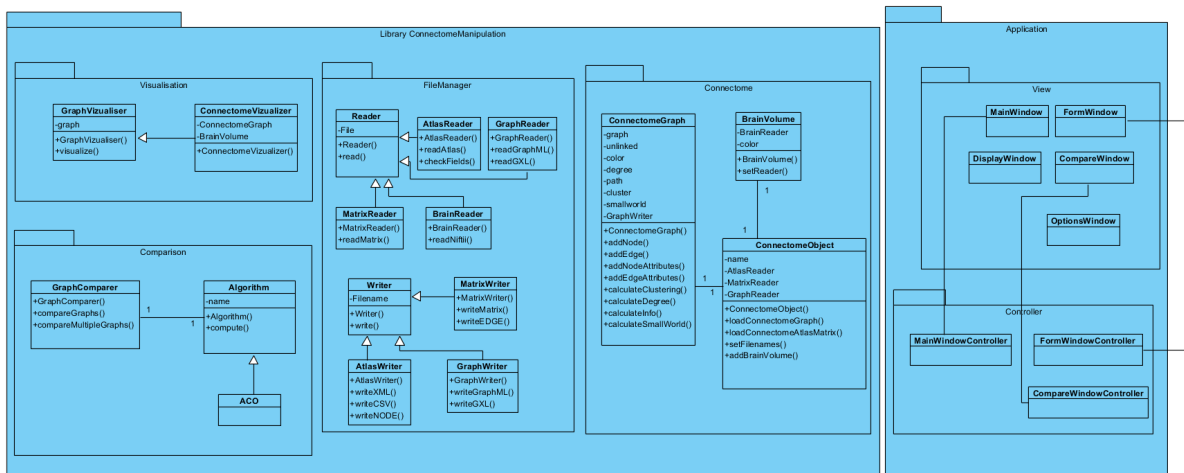


Figure 1 – Diagramme de classe final du projet de recherche et développement

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.

### 1.1 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
- Comparaison 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 faites et ont été laissées à une continuation du projet. Les classes concernées sont :

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

## 1.2 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.

## 2 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

## 3 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](#). 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](#). Il permet de tester la lecture d'atlas en XML.
- L'atlas `msdl_rois_labels.csv` et `msdl_matrix.txt` aussi extraits de [FSL](#). 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 brainigraph.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 [Brainigraph.org](#).

- Des graphes de connectomes de rats et de vers récupérés sur [Neurodata.io](#)

## 4 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 mesh3D 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.

## 5 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.

# 12

## Gestion de projet

### 1 Planification du projet

Une première planification du projet a été faite au cours du S9, une fois le projet déjà commencé. Cette planification a été faite à mi-novembre et recense donc la répartition de septembre à novembre en plus de planifier le reste du projet.

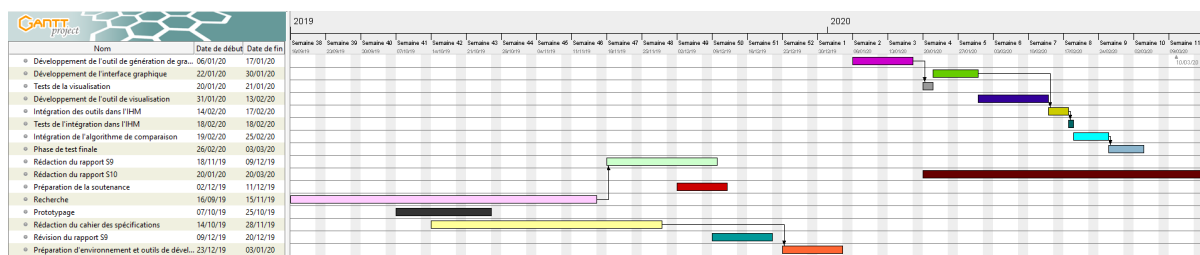


Figure 1 – Diagramme de Gantt du projet de recherche et développement

Étant le premier projet de cette envergure, la planification de la phase de développement a été très approximative : il est difficile d'estimer efficacement la durée de développement de chaque module, notamment par rapport au fait que ce sont des technologies et librairie avec lesquelles je n'ai pas d'expérience.

Ainsi, le diagramme de Gantt ci-dessus est purement à titre indicatif et n'a pour objectif que de décrire les différentes phases du projet et l'ordre de développement du projet.

### 2 Cycle de développement

Le projet va suivre un cycle de développement agile basé sur des sprints d'une à deux semaines en fonction du module en cours de développement.

Chaque phase de développement sera suivi d'une phase de test pour valider le fonctionnement du module et une phase de test finale sera effectuée pour vérifier l'interaction entre les modules.

Après le développement du premier module de génération de graphe, le planning de développement a été revu avec un nouvel objectif de développement : produire non seulement une

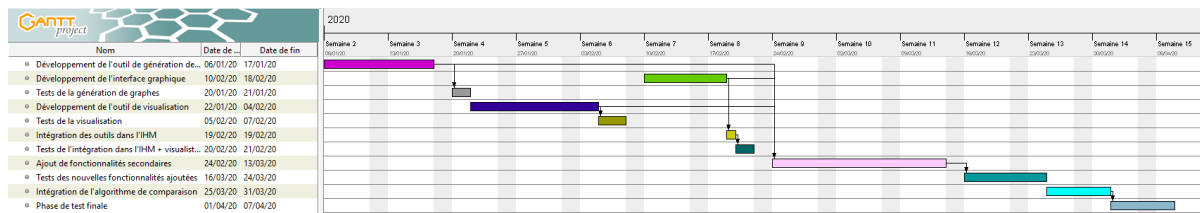


Figure 2 – Diagramme de Gantt final du développement

application de gestion de connectomes, mais aussi des classes permettant une utilisation plus large, en développant le début d'une bibliothèque de gestion de connectomes.

Le diagramme de Gantt a donc été revu de manière à implémenter cette nouvelle vision, notamment par la révision des priorités des différentes fonctionnalités et l'ajout d'une période de temps dédiée à améliorer ou ajouter des fonctionnalités qui n'étaient pas prévues dans le cahier des spécifications original.

# 13

## Documentation d'installation

### 1 Environnement de développement du projet

Le langage de programmation du projet est Python 3.6, et le projet a été développé à l'aide de Pycharm.

Plusieurs bibliothèques ont été utilisées lors du développement de ce projet :

- 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

Ces bibliothèques sont nécessaires pour que l'application fonctionne. L'application utilise comme point de départ le fichier « App.py » qui ouvre la fenêtre principale de l'application.

### 2 Exécutable et PyInstaller

La bibliothèque Pyinstaller [PyInstaller](#) permet de construire un fichier en .exe qui permet de lancer l'application Python. Pour créer cet exécutable, il faut utiliser la commande

*pyinstaller/path/to/App.py*

Cependant, étant donné la complexité du projet, certaines options sont nécessaires pour construire un exécutable ayant accès aux différentes bibliothèques nécessaires à l'exécution. L'utilisation d'un fichier .bat permettrait aussi de lancer l'application à la manière d'un exécutable.



# 14

## Documentation d'utilisation

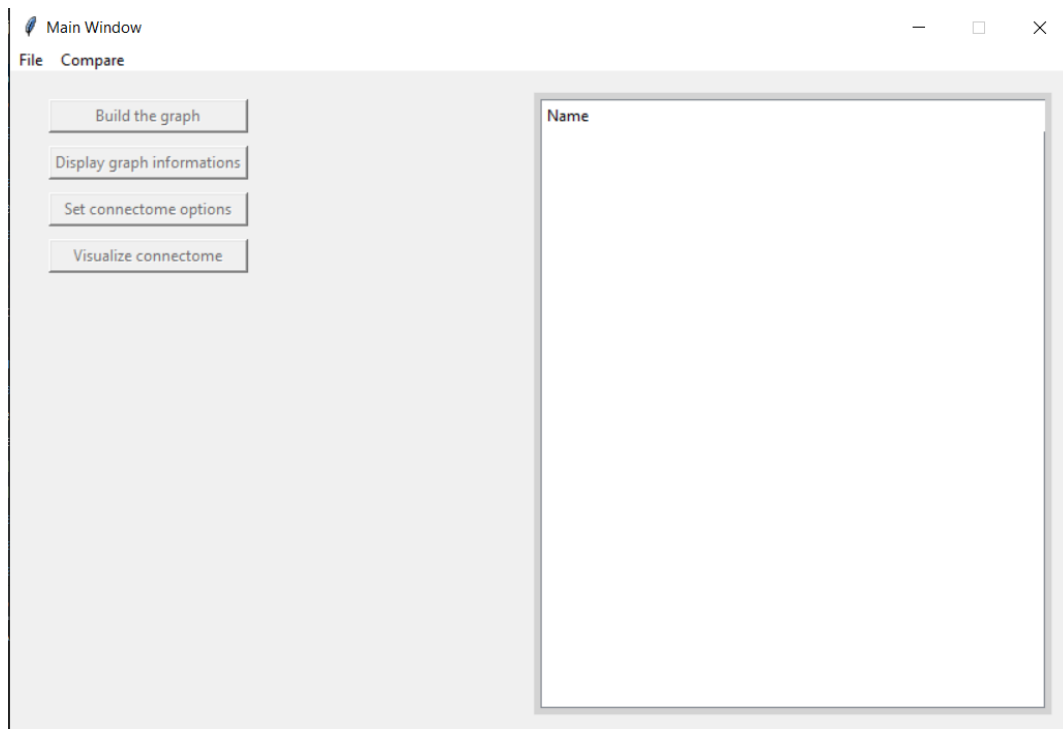
### 1 Présentation de l'application

L'application présentée est une application de manipulation de connectomes qui permet de construire, à partir de données de neuroimagerie médicales, des graphes représentant des connectomes et d'ensuite les manipuler.

L'objectif est de permettre l'utilisation des graphes pour représenter et étudier les connectomes. Les connectomes sont des objets complexes permettant de cartographier les connexions du cerveau, et une représentation sous forme de graphe permet d'utiliser tous les outils de la théorie des graphes pour étudier les connectomes.

Une partie de ce guide sera aussi dédiée à la construction des classes composant l'application qui permettent aussi de représenter des connectomes en dehors du cadre de l'application.

Plusieurs fonctionnalités sont disponibles avec les connectomes importés, et vont être décrites dans les parties suivantes.



**Figure 1** – Fenêtre principale de l'application

## 2 Utilisation de l'application

### 2.1 Créer un graphe de connectome

Pour pouvoir utiliser l'application, il est nécessaire dans un premier temps de créer un graphe avec les informations du connectome. Pour construire un graphe de connectome, il est nécessaire d'avoir à minima deux fichiers :

- Un atlas qui va permettre de construire les sommets du graphe
- Une matrice d'adjacence qui va permettre de construire les arêtes du graphe

Pour pouvoir créer un nouveau connectome, il faut cliquer dans le menu File → New :

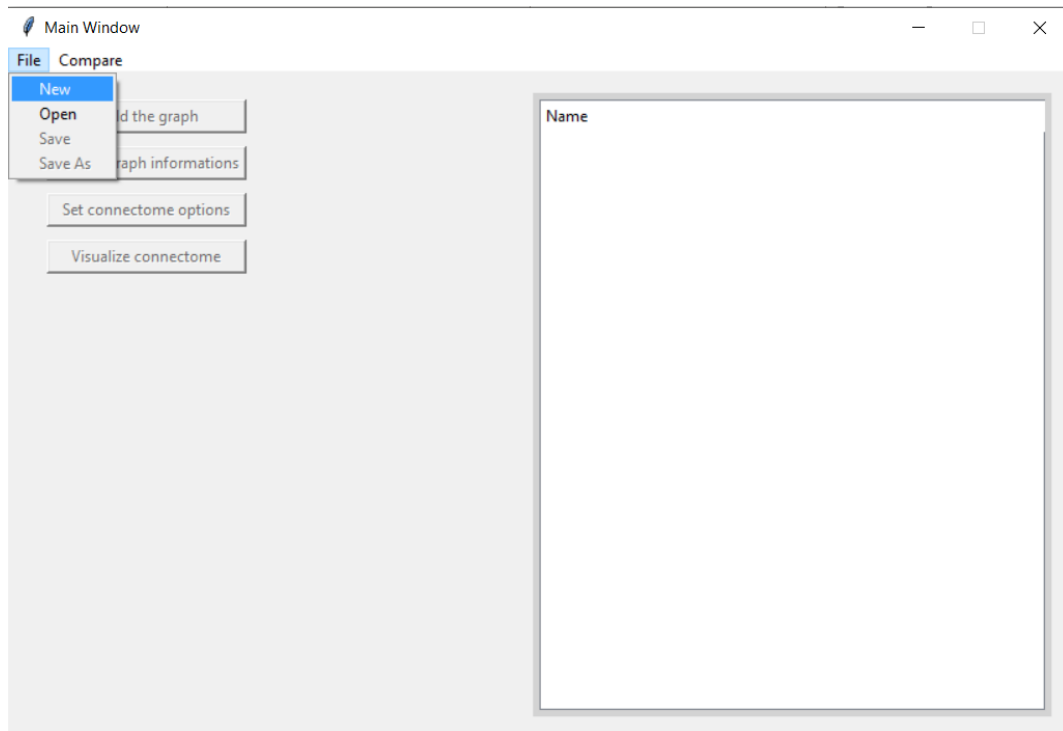


Figure 2 – Menu fichier de l'application

Ce qui va ouvrir la fenêtre de création d'un connectome :

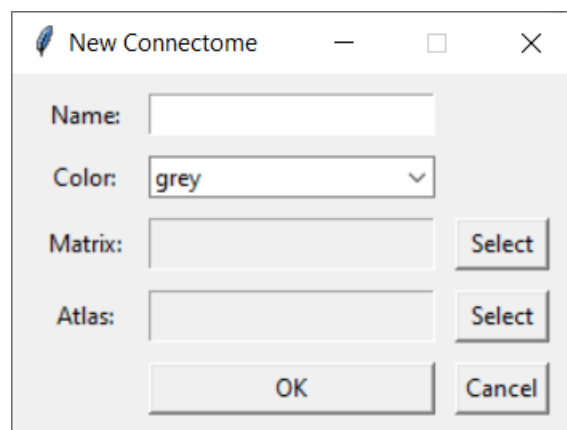


Figure 3 – Fenêtre de création de connectome

Dans cette fenêtre, on doit renseigner le nom du connectome, sa couleur et sélectionner des fichiers de matrices et d'atlas à importer.

L'application permet plusieurs types de fichier pour l'atlas et la matrice :

- L'atlas peut être au format .txt, .csv, .node ou .xml
- La matrice d'adjacence peut être au format .txt ou .edge.

Ces types de fichiers permettent de couvrir les atlas et matrices de logiciels tels que FSL ou BrainNet Viewer.

Par exemple, les données peuvent être de ces types :

x,y,z,name,net name
-53.28,-8.88,32.36,L Aud,Aud
53.47,-6.49,27.52,R Aud,Aud
1.18,-74.54,10.79,Striate,Striate
-45.8,-64.78,31.84,L DMN,DMN
-0.2,-55.21,29.87,Med DMN,DMN
-0.15,51.42,7.58,Front DMN,DMN
51.66,-59.34,28.88,R DMN,DMN
0.41,-91.05,1.58,Occ post,Occ post
-1.48,-27.93,61.5,Motor,Motor
40.1,20.96,44.72,R DLPFC,R V Att
37.83,55.49,1.22,R Front pol,R V Att
47.53,-52.42,43.06,R Par,R V Att
62.53,-32.99,-9.14,R Post Temp,R V Att
-0.91,-2.75,6.15,Basal,Basal
-41.66,-59.04,44.61,L Par,L V Att
-39.04,19.28,43.27,L DLPFC,L V Att
-40.08,50.65,0.81,L Front pol,L V Att
-29.39,-59.43,44.2,L IPS,D Att
31.6,-58.09,45.69,R IPS,D Att
-30.54,-85.14,9.1,L LOC,Vis Sec
-24.29,-74.28,-11.74,Vis,Vis Sec
33.4,-77.96,4.31,R LOC,Vis Sec
-28.17,46.32,21.56,D ACC,Saliency
-0.45,34.06,20.73,V ACC,Saliency
28.38,47.72,22.13,R A Ins,Saliency
-52.12,-17.92,13.28,L STS,Temporal

Figure 4 – Extrait de l'atlas msdl rois.csv

0.85	0.86	0.91	0.69	0.42	0.18	0.35	0.84	0.47	0.83	0.36	0.09	0.63	0.91
0.56	0.93	0.53	0.76	0.43	0.73	0.89	0.74	0.56	0.61	0.45	0.43	0.54	0.70
0.93	0.98	0.11	0.43	0.12	0.37	0.45	0.95	0.27	0.57	0.39	0.26	0.44	0.73
0.70	0.86	0.83	0.66	0.02	0.84	0.41	0.03	0.75	0.33	0.78	0.30	0.29	0.23
0.58	0.79	0.34	0.11	0.29	0.73	0.22	0.36	0.50	0.46	0.73	0.42	0.50	0.58
0.82	0.51	0.29	0.93	0.32	0.57	0.13	0.66	0.65	0.71	0.43	0.12	0.76	0.81
0.88	0.18	0.75	0.19	0.65	0.18	0.31	0.28	0.31	0.88	0.69	0.50	0.76	0.40
0.99	0.40	0.01	0.27	0.96	0.96	0.73	0.23	0.14	0.72	0.95	0.71	0.58	0.99
0.00	0.13	0.05	0.80	0.94	0.27	0.78	0.71	0.48	0.02	0.78	0.24	0.75	0.09
0.87	0.03	0.67	0.49	0.46	0.92	0.69	0.62	0.36	0.67	0.71	0.79	0.65	0.32
0.61	0.94	0.60	0.77	0.24	0.22	0.01	0.59	0.79	0.44	0.11	0.07	0.12	0.51
0.99	0.30	0.53	0.40	0.76	0.37	0.84	0.66	0.78	0.44	0.39	0.39	0.50	0.06
0.53	0.30	0.73	0.27	0.76	0.09	0.92	0.05	0.67	0.12	0.59	0.00	0.35	0.73
0.48	0.33	0.71	0.04	0.74	0.64	0.77	0.35	0.13	0.81	0.46	0.22	0.09	0.56
0.80	0.47	0.78	0.67	0.74	0.18	0.04	0.45	0.02	0.32	0.05	0.00	0.15	0.53
0.23	0.65	0.29	0.43	0.11	0.05	0.38	0.24	0.56	0.25	0.23	0.19	0.20	0.83
0.50	0.03	0.69	0.45	0.68	0.72	0.70	0.72	0.30	0.34	0.83	0.14	0.67	0.86
0.90	0.84	0.56	0.61	0.46	0.35	0.73	0.86	0.94	0.38	0.02	0.27	0.43	0.79
0.57	0.56	0.40	0.06	0.21	0.66	0.22	0.28	0.98	0.55	0.86	0.17	0.69	0.32
0.85	0.85	0.06	0.32	0.10	0.38	0.27	0.73	0.29	0.56	0.08	0.14	0.26	0.45
0.74	0.35	0.78	0.77	0.82	0.63	0.67	0.14	0.80	0.40	0.67	0.60	0.01	0.75
0.59	0.45	0.34	0.70	0.18	0.02	0.48	0.84	0.90	0.40	0.50	0.90	0.53	0.11
0.25	0.05	0.61	0.13	0.16	0.91	0.62	0.14	0.60	0.52	0.22	0.94	0.28	0.11
0.67	0.18	0.74	0.13	0.67	0.80	0.24	0.59	0.88	0.66	0.57	0.22	0.95	0.27
0.08	0.66	0.10	0.09	0.89	0.75	0.18	0.37	0.94	0.95	0.12	0.48	0.91	0.52

Figure 5 – Extrait de la matrice Brodmann82.edge

Une fois les fichiers renseignés, il est possible de choisir des options pour la construction de connectome.

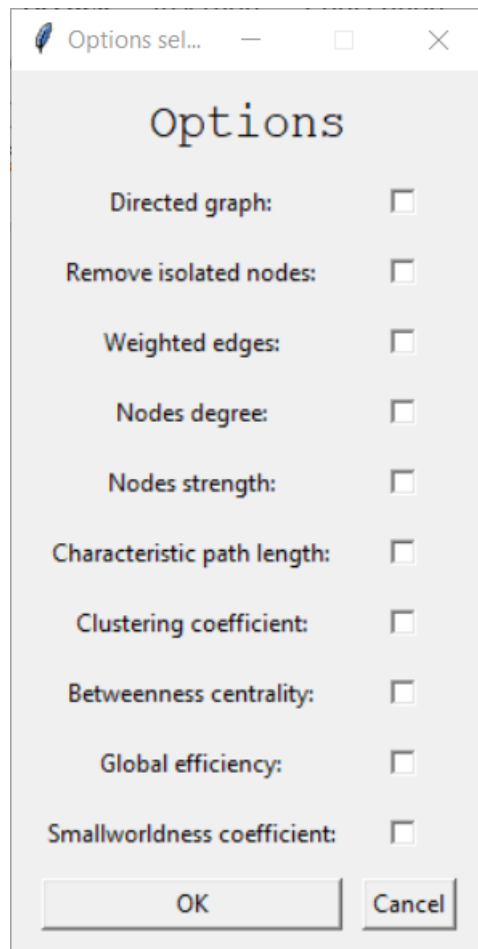


Figure 6 – Options de constructions de connectome

Les options permettent de choisir le type de graphe construit et les potentiels attributs supplémentaires qu'il est possible de calculer. Les options sont les suivantes :

- Directed graph : si possible, la matrice d'adjacence est lue de manière à construire un graphe orienté
- Remove isolated nodes : supprime du graphe les sommets sans arêtes s'ils existent
- Weighted edges : si la matrice d'adjacence possède des poids, ajoute un poids aux arêtes
- Nodes degree : calcule le degré des sommets
- Nodes strength : calcule le degré pondéré des sommets (nécessite des arêtes pondérées)
- Characteristic path length : calcule la Moyenne des plus courts chemins du graphe
- Clustering coefficient : calcule le coefficient de clustering des sommets
- Betweenness centrality : calcule la centralité des sommets
- Global efficiency : calcule l'efficacité globale du graphe (nécessite un graphe non-dirigé)
- Smallworldness coefficient : calcule le coefficient de réseau « petit-monde » du graphe (nécessite de ne pas avoir de sommets isolés et un graphe non-dirigé)

Le connectome est ensuite ajouté à la liste des connectomes disponibles et est automatiquement sélectionné. Il suffit ensuite d'appuyer sur le bouton « Build the graph » pour construire le graphe correspondant au connectome construit.

Une fenêtre apparaîtra pour demander à l'utilisateur quel type de graphe il veut construire. Plusieurs options sont disponibles :

- Default : construit le graphe entier
- Left : construit un sous-graphe correspondant à l'hémisphère gauche du connectome
- Right : construit un sous-graphe correspondant à l'hémisphère droit du connectome

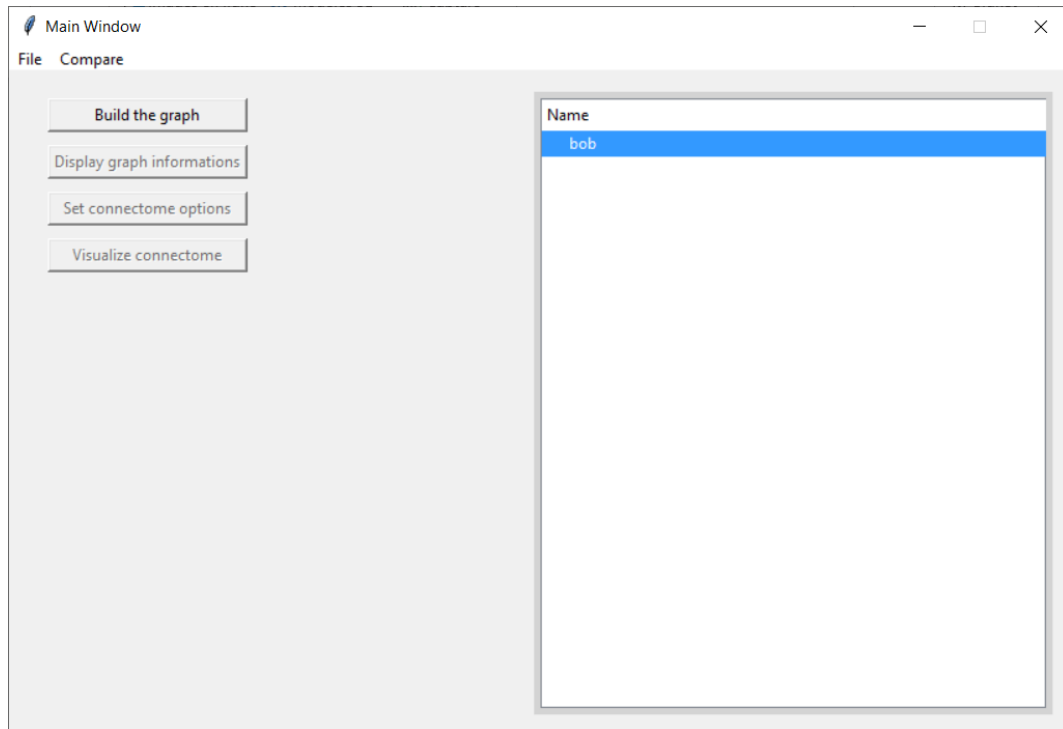


Figure 7 – Connectome ajouté à la liste

## 2.2 Importer un graphe préexistant

Il est aussi possible d'importer un graphe déjà construit pour le réutiliser, ou d'importer un graphe de connectome d'une application tierce. Il suffit pour cela d'aller dans le menu File → Open et de choisir le graphe à importer. Actuellement, le seul format d'import supporté est le format graphml.

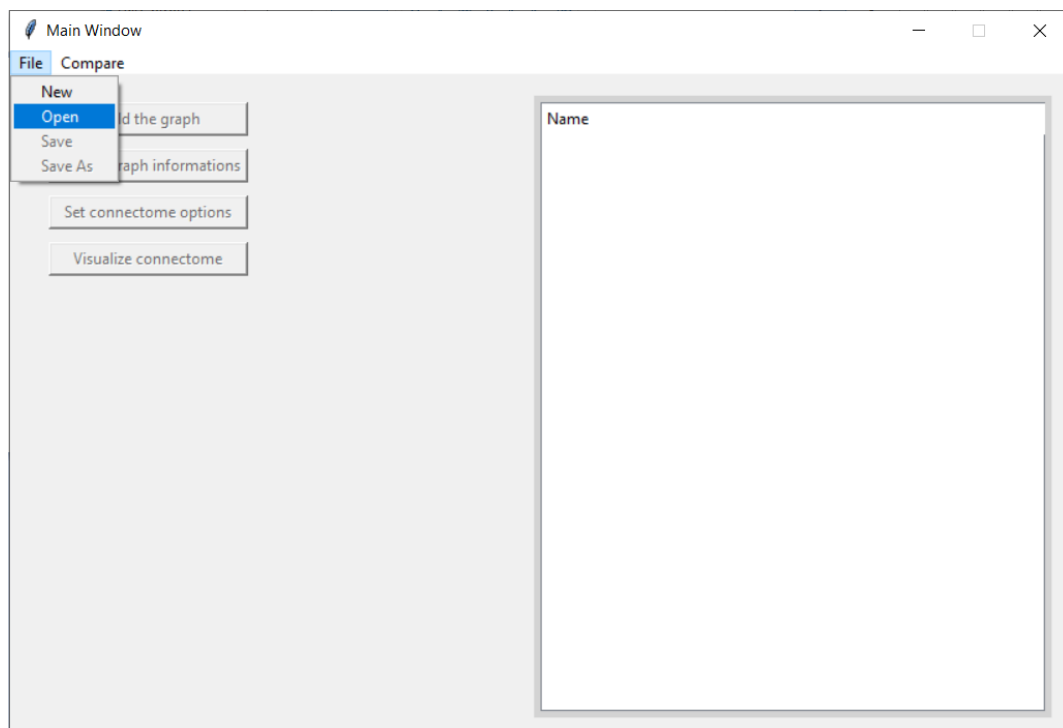


Figure 8 – Import d'un graphe graphml

Il suffit ensuite de choisir le fichier graphml à importer et un graphe correspondant au contenu du fichier est construit.

### 2.3 Modifier les informations du graphe

Il est possible d'ajouter aux graphes importés ou créer des informations supplémentaires en tant qu'attributs de sommets. On va pour cela réutiliser les options présentées plus haut pour ajouter des attributs calculables ou modifier le type de graphe si c'est un graphe créé à partir d'une matrice et d'un atlas (on peut décider de recalculer le graphe en tant que graphe orienté par exemple. On utilise pour cela le bouton "Set connectome options" qui ouvre la fenêtre d'option montrée plus haut et permet de sélectionner de nouvelles options. Le graphe est ensuite reconstruit en prenant compte des nouvelles options.

Il est aussi possible à tout moment de reconstruire le graphe avec le bouton "Build the graph" pour passer d'un graphe entier à un hémisphère ou inversement.

On peut afficher des informations sur le graphe en cliquant sur le bouton "Display informations".

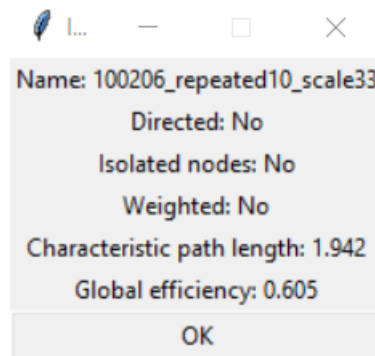
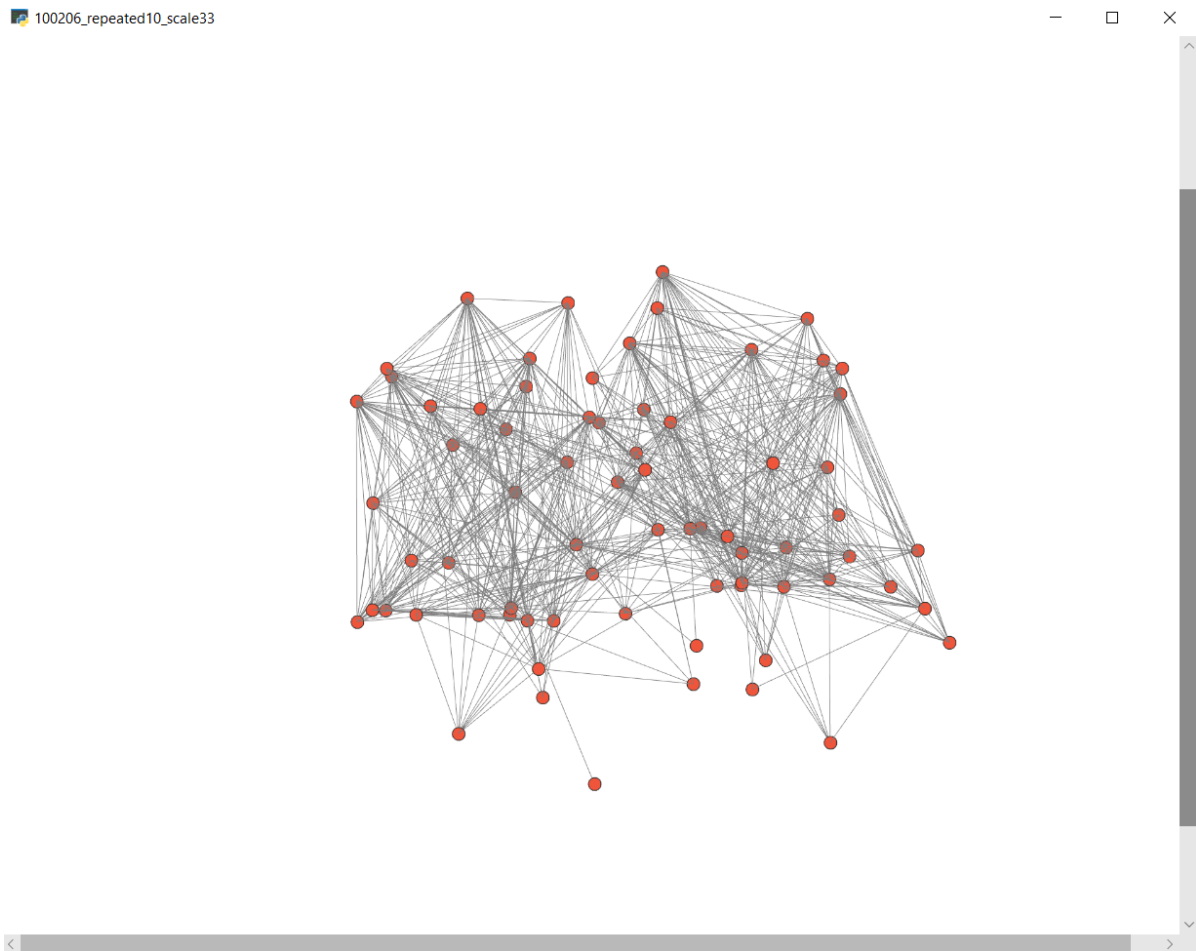


Figure 9 – Affichage des informations sur le graphe

### 2.4 Visualiser le connectome

Il est possible d'afficher une représentation 3D du connectome à l'aide du bouton « Visualize connectome ». La visualisation va se faire sur le connectome actuellement sélectionné. Cette opération peut prendre quelques instants en fonction de la complexité du graphe.

Si le graphe à visualiser est un graphe entier, une fenêtre d'option de visualisation va être affichée, permettant de choisir de visualiser un sous graphe, de l'hémisphère droit ou gauche. Si le graphe à visualiser est déjà un sous-graphe, l'affichage correspondant sera fait.



**Figure 10** – *Visualisation d'un graphe de connectome*

Des informations sur les sommets sont affichées au passage de la souris sur le sommet.

## 2.5 Sauvegarder des graphes de connectomes

Il est possible de sauvegarder les graphes créés ou modifiés dans le menu File → Save As. Le graphe est sauvegardé sous format graphml, et il est possible pour un graphe importé de sauvegarder directement sous le même nom avec File → Save.

## 3 Bibliothèque de manipulation de connectomes

Dans cette partie, on va s'intéresser à la construction du code et la structure des classes représentant les connectomes.

Cette application est une application en Python de gestion de connectomes et les objets utilisés par cette application font partie d'une bibliothèque de gestion de connectomes qui permet plusieurs fonctionnalités.

Cette partie ne sert pas de documentation au code, mais seulement d'explication de la structure du code pour comprendre le fonctionnement des objets. Les principales classes seront présentées.



### 3.1 ConnectomeObject

Les connectomes sont représentés par une classe `ConnectomeObject`. Cette classe possède plusieurs attributs, dont le `ConnectomeGraph` qui est une représentation sous forme de graphe du connectome. La classe `ConnectomeObject` est l'objet principal de la bibliothèque qui va être utilisée par les autres classes. Un `ConnectomeObject` possède un certain nombre de `Reader` et `Writer` qui permettent de charger des données ou écrire les données du connectomes dans les différents fichiers :

- `MatrixReader` permet de lire une matrice d'adjacence
- `AtlasReader` permet de lire un atlas
- `GraphReader` permet de lire un fichier de graphe
- `GraphWriter` permet d'écrire le connectome dans un fichier de graphe

### 3.2 ConnectomeGraph

La classe `ConnectomeGraph` permet de représenter le connectome concerné sous la forme d'un graphe. Pour ce faire, un des attributs de la classe est l'attribut « `graph` » qui est un graphe `networkx`.

La classe `ConnectomeGraph` permet aussi d'effectuer des calculs et d'ajouter des attributs au graphe.

# 15

## Cahier de tests

Pour effectuer des tests pertinents sur les connectomes, des données d'entrées ont été récupérées depuis différentes sources :

- Des matrices d'adjacence Brodmann82.edge et atlas Brodmann82.node de Brain Net Viewer
- Des atlas Harvard-Oxford.xml et MSDL.csv de FSL
- Des graphes de connectomes animaux depuis le site [Neurodata.io](http://Neurodata.io)
- Des jeux de graphes de connectomes depuis le site [Braingraph.org](http://Braingraph.org)
- Une matrice d'adjacence et un atlas fourni par l'hôpital de Tours

### 1 Tests de graphes

Les tests de graphes ont pour objectif de tester les fonctionnalités sur les graphes de connectomes. Ces tests couvrent la lecture, l'écriture de graphe de connectomes ainsi que certains calculs d'attributs et opération sur les graphes telles que la comparaison et le changement d'options.

#### 1.1 Test hospital

Teste la création d'un graphe à partir d'une matrice et d'un atlas au format txt. Écrit ensuite le graphe créé dans un fichier graphml. Il est possible de comparer l'input avec le fichier graphml d'output pour vérifier la correspondance des données du graphe.

**Assert :**

« hospital.graphml » n'existe pas au début et existe à la fin du test

Résultat : un fichier « hospital.graphml » est créé de manière quasiment instantanée

#### 1.2 Test edge node

Teste la création d'un graphe à partir d'une matrice au format edge et d'un atlas au format node. Écrit ensuite le graphe créé dans un fichier graphml. Il est possible de comparer l'input avec le fichier graphml d'output pour vérifier la correspondance des données du graphe.

**Assert :**

« brodmann82.graphml » n'existe pas au début et existe à la fin du test

Résultat : un fichier « brodmann82.graphml » est créé de manière quasiment instantanée

**1.3 Test xml**

Teste la création d'un graphe à partir d'une matrice au format txt et d'un atlas au format xml. Écrit ensuite le graphe créé dans un fichier graphml. Il est possible de comparer l'input avec le fichier graphml d'output pour vérifier la correspondance des données du graphe.

**Assert :**

« harvardoxford.graphml » n'existe pas au début et existe à la fin du test

Résultat : un fichier « harvardoxford.graphml » est créé de manière quasiment instantanée

**1.4 Test csv**

Teste la création d'un graphe à partir d'une matrice au format txt et d'un atlas au format csv. Écrit ensuite le graphe créé dans un fichier graphml. Il est possible de comparer l'input avec le fichier graphml d'output pour vérifier la correspondance des données du graphe.

**Assert :**

« msdl.graphml » n'existe pas au début et existe à la fin du test

Résultat : un fichier « msdl.graphml » est créé de manière quasiment instantanée

**1.5 Test from graph brodmann**

Teste la création d'un graphe à partir d'un fichier graphml. Écrit ensuite le graphe créé dans un nouveau fichier graphml.

**Assert :**

« brodmann82.graphml » existe au début et est équivalent à l'output

« brodmann82\_from\_graphml.graphml »

Résultat : un fichier « brodmann82\_from\_graphml.graphml » est créé de manière quasiment instantanée

**1.6 Test from graph hospital**

Teste la création d'un graphe à partir d'un fichier graphml. Écrit ensuite le graphe créé dans un nouveau fichier graphml.

**Assert :**

« hospital.graphml » existe au début et est équivalent à l'output

« hospital\_from\_graphml.graphml »

Résultat : un fichier « hospital\_from\_graphml.graphml » est créé de manière quasiment instantanée

### 1.7 Test from graph harvardoxford

Teste la création d'un graphe à partir d'un fichier graphml. Écrit ensuite le graphe créé dans un nouveau fichier graphml.

**Assert :**

« harvardoxford.graphml » existe au début et est équivalent à l'output  
« harvardoxford\_from\_graphml.graphml »

Résultat : un fichier « harvardoxford\_from\_graphml.graphml » est créé de manière quasiment instantanée

### 1.8 Test from graph msdl

Teste la création d'un graphe à partir d'un fichier graphml. Écrit ensuite le graphe créé dans un nouveau fichier graphml.

**Assert :**

« msdl.graphml » existe au début et est équivalent à l'output  
« msdl\_from\_graphml.graphml »

Résultat : un fichier « msdl\_from\_graphml.graphml » est créé de manière quasiment instantanée

### 1.9 Test calculations

Teste les calculs des différents métriques calculables sur les graphes de connectomes. Au moment de l'écriture du cahier de test, les métriques calculées sont les suivantes :

- Degré des sommets
  - Force des sommets
  - Coefficient de clustering
  - Centralité
  - Plus court chemin moyen
  - Efficacité du graphe
- Ces calculs sont effectués sur un graphe simple où les calculs des fonctions sont comparés avec des valeurs précalculées pour ce graphe.

**Assert :**

Degré des sommets précalculé égal au degré des sommets obtenu  
Force des sommets précalculée égale à la force des sommets obtenue  
Coefficient de clustering précalculé égal au coefficient de clustering obtenu  
Centralité précalculée égale à la centralité obtenue  
Plus court chemin moyen précalculé égal au plus court chemin moyen obtenu  
Efficacité du graphe précalculée égale à l'efficacité du graphe obtenue

Résultat : les valeurs précalculées sont égales aux valeurs obtenues par le biais des fonctions de calcul

### 1.10 Test Subgraph

Teste la création de sous graphes représentant l'hémisphère gauche et droit du connectome. Cependant, le graphe total n'est pas la somme des deux sous-graphes, puisque l'on perd les arêtes entre les sommets des deux hémisphères.

**Assert :**

Le fichier d'entrée existe  
 Le graphe créé à partir de ce fichier n'est pas nul  
 Le sous-graphe gauche n'est pas nul  
 Le sous-graphe droit n'est pas nul  
 L'union des sous-graphe n'est pas nul

Résultat : On obtient deux sous graphes correspondant aux hémisphère qui, combinés, correspondent au graphe originel sans les connections entre les deux hémisphères

**1.11 Test SymmetricDifference**

Teste de l'algorithme SymmetricDifference de networkx sur deux graphes similaires. Cet algorithme crée un graphe composé des arêtes qui sont présentes dans un des deux graphes mais pas les deux.

**Assert :**

Les graphes lus dans les fichiers ne sont pas nuls  
 Le graphe résultant des arêtes présente dans un des deux graphes n'est pas nul  
 Les graphes comparés et le résultat de la comparaison sont visualisables

Résultat : On obtient un graphe composé des arêtes présentes dans seulement un des deux graphes

**2 Tests de visualisation**

Les tests de visualisation ont pour but de tester la fonction de visualisation sur les différents types de données d'entrée.

**2.1 Test brodmann82view**

Teste la visualisation d'un graphe à partir d'une matrice au format edge et d'un atlas au format node. On peut ensuite vérifier les attributs à l'œil nu.

**Assert :**

La fonction de visualisation retourne True

Résultat : la visualisation du graphe Brodmann82 est affichée

**2.2 Test hospitalview**

Teste la visualisation d'un graphe à partir d'une matrice au format txt et d'un atlas au format txt. On peut ensuite vérifier les attributs à l'œil nu.

**Assert :**

La fonction de visualisation retourne True

Résultat : la visualisation du graphe Hospital est affichée

### 2.3 Test msdlview

Teste la visualisation d'un graphe à partir d'une matrice au format txt et d'un atlas au format csv. On peut ensuite vérifier les attributs à l'œil nu.

**Assert :**

La fonction de visualisation retourne True

Résultat : la visualisation du graphe MSDL est affichée

### 2.4 Test harvardoxfordview

Teste la visualisation d'un graphe à partir d'un fichier graphml. On peut ensuite vérifier les attributs à l'œil nu.

**Assert :**

La fonction de visualisation retourne True

Résultat : la visualisation du graphe HarvardOxford est affichée

### 2.5 Test visualizeLeft

Teste l'option « left » de la visualisation qui vise à ne visualiser que l'hémisphère gauche du cerveau. On peut ensuite vérifier les attributs à l'œil nu.

**Assert :**

La visualisation du graphe d'entrée en option « left » fonctionne

Résultat : la visualisation du sous graphe hémisphère gauche de Brodmann82 est affichée

### 2.6 Test visualizeRight

Teste l'option « right » de la visualisation qui vise à ne visualiser que l'hémisphère gauche du cerveau. On peut ensuite vérifier les attributs à l'œil nu.

**Assert :**

La visualisation du graphe d'entrée en option « right » fonctionne

Résultat : la visualisation du sous graphe hémisphère droit de Brodmann82 est affichée



# Webographie

- [WWW1] *Brain Graphs*. URL : <http://braph.org/manual/brain-graphs/> (visité le 28/11/2019).
- [WWW2] Timothy Busbice. *Building Artificial Connectomes*. July 26, 2016. URL: <https://www.oreilly.com/ideas/building-artificial-connectomes> (visited on 12/07/2019).
- [WWW3] *Connectome cérébral humain – Groupe d’imagerie neurofonctionnelle (GIN-IMN)*. URL : <http://www.gin.cnrs.fr/fr/recherche/axe3/> (visité le 08/12/2019).
- [WWW4] *Connectomes*. URL : <https://neurodata.io/project/connectomes/> (visité le 19/12/2019)■
- [WWW5] *HCP Aging - Connectome - Publications*. URL : <https://www.humanconnectome.org/study/hcp-lifespan-aging> (visité le 02/12/2019).
- [WWW6] *Sex Differences in the Structural Connectome of the Human Brain*. URL : <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3896179/> (visité le 28/11/2019).

# Bibliographie

- [1] Adriana DI MARTINO, David O'CONNOR, Bosi CHEN, Kaat ALAERTS, Jeffrey S. ANDERSON, Michal ASSAF, Joshua H. BALSTERS, Leslie BAXTER, Anita BEGGIATO, Sylvie BERNAERTS et al. « Enhancing Studies of the Connectome in Autism Using the Autism Brain Imaging Data Exchange II ». In : *Scientific Data* 4 (14 mar. 2017). ISSN : 2052-4463. DOI : [10.1038/sdata.2017.10](https://doi.org/10.1038/sdata.2017.10). PMID : 28291247. URL : <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5349246/> (visité le 02/12/2019).
- [2] Emily S. FINN, Xilin SHEN, Dustin SCHEINOST, Monica D. ROSENBERG, Jessica HUANG, Marvin M. CHUN, Xenophon PAPADEMETRIS et R. Todd CONSTABLE. « Functional Connectome Fingerprinting : Identifying Individuals Based on Patterns of Brain Connectivity ». In : *Nature neuroscience* 18.11 (nov. 2015), p. 1664-1671. ISSN : 1097-6256. DOI : [10.1038/nn.4135](https://doi.org/10.1038/nn.4135). PMID : 26457551. URL : <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5008686/> (visité le 04/12/2019).
- [3] Olfa Sammoud, Christine Solnon, and Khaled Ghédira. "Ant Algorithm for the Graph Matching Problem". In: *Evolutionary Computation in Combinatorial Optimization*. Ed. by Günther R. Raidl and Jens Gottlieb. Red. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, et al. Vol. 3448. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 213–223. ISBN: 978-3-540-25337-2 978-3-540-31996-2. DOI: [10.1007/978-3-540-31996-2\\_20](https://doi.org/10.1007/978-3-540-31996-2_20). URL: [http://link.springer.com/10.1007/978-3-540-31996-2\\_20](http://link.springer.com/10.1007/978-3-540-31996-2_20) (visited on 11/28/2019).
- [4] Xilin SHEN, Emily S. FINN, Dustin SCHEINOST, Monica D. ROSENBERG, Marvin M. CHUN, Xenophon PAPADEMETRIS et R Todd CONSTABLE. « Using Connectome-Based Predictive Modeling to Predict Individual Behavior from Brain Connectivity ». In : *Nature protocols* 12.3 (mar. 2017), p. 506-518. ISSN : 1754-2189. DOI : [10.1038/nprot.2016.178](https://doi.org/10.1038/nprot.2016.178). PMID : 28182017. URL : <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5526681/> (visité le 28/11/2019).
- [5] Ni SHU, Ying LIANG, He LI, Junying ZHANG, Xin LI, Liang WANG, Yong HE, Yongyan WANG et Zhanjun ZHANG. « Disrupted Topological Organization in White Matter Structural Networks in Amnesic Mild Cognitive Impairment : Relationship to Subtype ». In : *Radiology* 265.2 (1<sup>er</sup> nov. 2012), p. 518-527. ISSN : 0033-8419. DOI : [10.1148/radiol.12112361](https://doi.org/10.1148/radiol.12112361). URL : <https://pubs.rsna.org/doi/full/10.1148/radiol.12112361> (visité le 28/11/2019).



- [6] Olaf Sporns, Giulio Tononi, and Rolf Kötter. “The Human Connectome: A Structural Description of the Human Brain”. In: *PLOS Computational Biology* 1.4 (Sept. 30, 2005), e42. ISSN: 1553-7358. DOI: [10.1371/journal.pcbi.0010042](https://doi.org/10.1371/journal.pcbi.0010042). URL: <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.0010042> (visited on 12/18/2019).
- [7] Jinhui Wang, Xinian Zuo, Zhengjia Dai, Mingrui Xia, Zhilian Zhao, Xiaoling Zhao, Jianping Jia, Ying Han, and Yong He. “Disrupted Functional Brain Connectome in Individuals at Risk for Alzheimer’s Disease”. In: *Biological Psychiatry*. Disturbances in the Connectome and Risk for Alzheimer’s Disease 73.5 (Mar. 1, 2013), pp. 472–481. ISSN: 0006-3223. DOI: [10.1016/j.biopsych.2012.03.026](https://doi.org/10.1016/j.biopsych.2012.03.026). URL: <http://www.sciencedirect.com/science/article/pii/S0006322312003034> (visited on 11/28/2019).
- [8] Van J. Wedeen, Patric Hagmann, Wen-Yih Isaac Tseng, Timothy G. Reese, and Robert M. Weisskoff. “Mapping Complex Tissue Architecture with Diffusion Spectrum Magnetic Resonance Imaging”. In: *Magnetic Resonance in Medicine* 54.6 (2005), pp. 1377–1386. ISSN: 1522-2594. DOI: [10.1002/mrm.20642](https://doi.org/10.1002/mrm.20642). URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.20642> (visited on 12/18/2019).
- [9] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner. “The Structure of the Nervous System of the Nematode *Caenorhabditis Elegans*”. In: *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences* 314.1165 (Nov. 12, 1986), pp. 1–340. ISSN: 0962-8436. DOI: [10.1098/rstb.1986.0056](https://doi.org/10.1098/rstb.1986.0056). pmid: [22462104](https://pubmed.ncbi.nlm.nih.gov/22462104/).

# Glossaire

**Ant Colony Optimization** Algorithme métaheuristique utilisé pour la résolution de problèmes d'optimisation. La comparaison de graphe est considérée comme un problème d'optimisation, donc on peut y appliquer l'ACO. Le principe de fonctionnement de l'ACO est principalement basé sur le comportement d'une colonie de fourmis auquel on ajoute des paramètres pour améliorer la performance. [3](#), [5](#), [12](#), [16](#), [26](#)

**Connectome** Le connectome est une manière de représenter les connexions entre les différentes régions du cerveau. Il existe plusieurs manières de représenter les connexions du cerveau, de manière fonctionnelle ou structurelle. La construction d'un connectome structurel se fait à l'aide d'IRM de diffusion qui permet d'étudier la diffusion aléatoire des molécules d'eau, et donc la densité et l'organisation des tissus. La construction d'un connectome fonctionnel se fait avec des données d'IRM fonctionnelle qui enregistre les variations de concentration de déoxy-hémoglobine dans le sang. Ces variations permettent de montrer l'activation des régions du cerveau et donc de construire une carte des connexions du cerveau au cours de la cognition. Un connectome peut être représenté à l'aide d'un graphe du cerveau utilisant les régions en tant que sommets et les connexions entre ces régions en tant qu'arêtes. Les arêtes porteront les informations sur les connexions, leur nature (fonctionnelle ou structurelle) ainsi que leur force ou la caractéristique concernée. [8](#)

**IRM** L'IRM est un examen médical qui permet d'avoir des vues en 2D ou 3D de l'intérieur du corps. Elle permet une étude avec grande précision en faisant des images en coupes de différents plans qui permettent de reconstruire en 3D la structure analysée. Le principe de fonctionnement est basé sur la résonance magnétique nucléaire, c'est-à-dire qu'avec un champ magnétique auquel on applique des champs oscillants plus faibles, on peut générer un signal électromagnétique mesurable. En neuro-imagerie, l'IRM permet d'étudier les tissus mous avec des contrastes élevés. [9](#), [14](#)

**NiFTI** Le format de fichier NiFTI est un format de fichier destiné à la neuroimagerie. Le 4D indique que le fichier utilisé est un fichier contenant des informations de neuroimagerie dans un environnement 3D en fonction du temps. [9](#), [14](#), [30](#)

**Regions Of Interest** Une région d'intérêt est un échantillon d'un jeu de données qui est isolé pour une étude. Dans le cadre de la neuro-imagerie médicale, les ROI sont les régions du cerveau définies pour créer les nœuds du connectome. Chaque ROI correspond à un nœud. Les différentes régions sont ensuite reliées entre elles par des arêtes représentant

les connexions entre les régions. En général, la séparation en région suit la séparation anatomique du cerveau. 9, 14

**tractogramme** Un tractogramme est une représentation sous formes de fibres virtuelles des connexions anatomiques entre les différentes régions du cerveau. Chaque fibre représente plusieurs milliers ou dizaines de milliers d'axone dans un réseau tridimensionnel.. 9

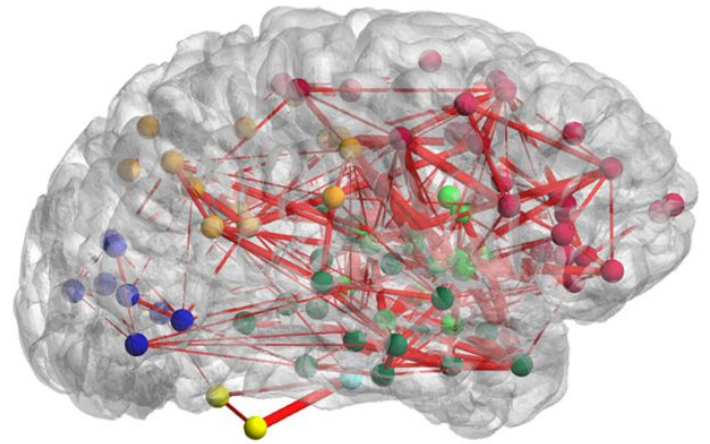
# Visualisation et comparaison de connectomes à l'aide de graphes

Clément Condette

Encadrement : Nicolas Monmarché, Kieu Diem Ho et Jean-Yves Ramel

## Connectomes

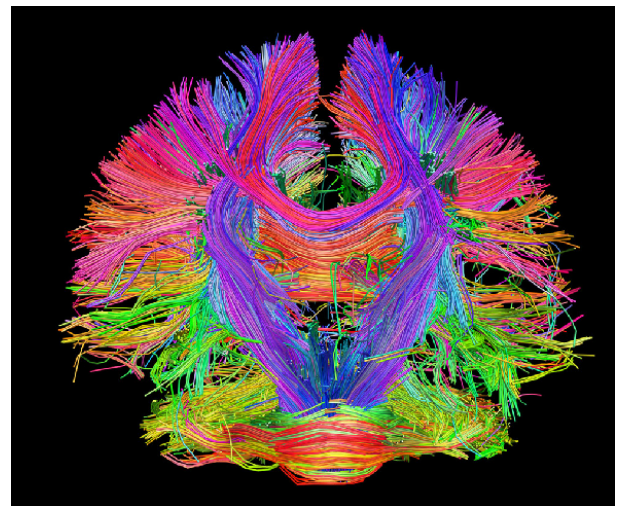
Un connectome est une carte des connexions du cerveau. On peut les représenter sous forme de graphe !



## Intérêt

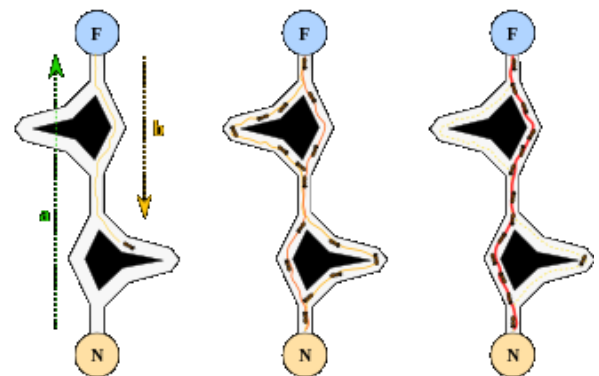
Il y en a plein :

- On peut comparer les connectomes pour étudier des maladies comme Alzheimer ou Parkinson
- En théorie, une IA utilisant un connectome modélisé parfaitement aurait la même personnalité que la personne dont il provient



## Étude

On peut utiliser la théorie des graphes pour étudier les connectomes !  
Des algorithmes bioinformatiques comme l'Ant Colony Optimization qui imite le comportement des fourmis par exemple



*Ant Colony Optimization*

# Visualisation et comparaison de connectomes à l'aide de graphes

Clément Condette

Encadrement : Nicolas Monmarché, Kieu Diem Ho et Jean-Yves Ramel

## Connectomes

Un connectome est une carte des connexions du cerveau. On peut les représenter sous forme de graphe !

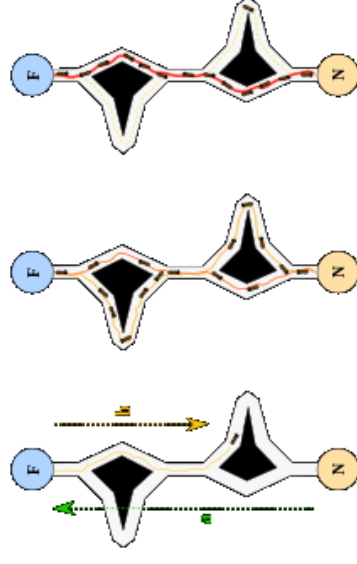
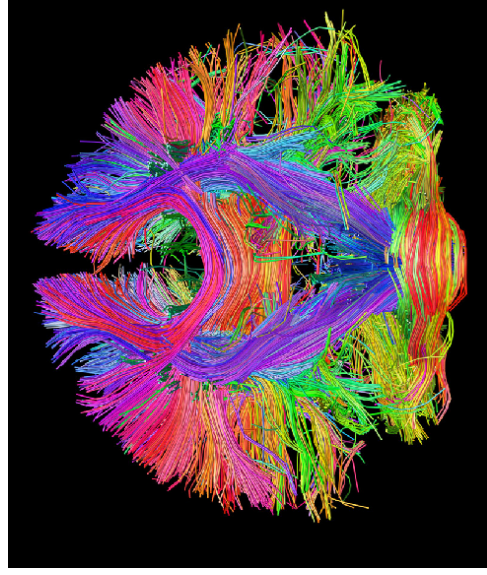
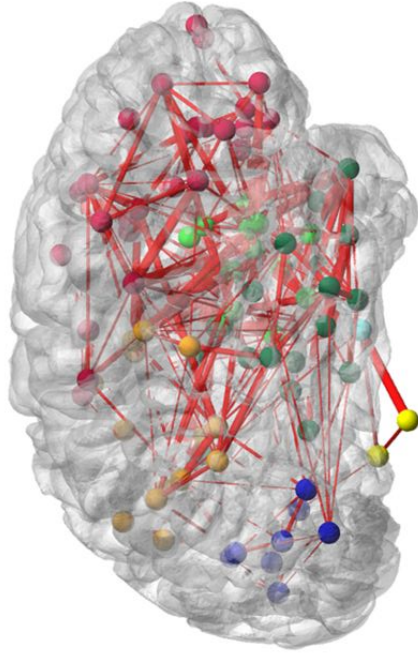
## Intérêt

Il y en a plein :

- On peut comparer les connectomes pour étudier des maladies comme Alzheimer ou Parkinson
- En théorie, une IA utilisant un connectome modélisé parfaitement aurait la même personnalité que la personne dont il provient

## Étude

On peut utiliser la théorie des graphes pour étudier les connectomes ! Des algorithmes bioinformatiques comme l'Ant Colony Optimization qui imite le comportement des fourmis par exemple



*Ant Colony Optimization*

# Visualisation et comparaison de connectomes à l'aide de graphes

## Résumé

Pour mieux comprendre le fonctionnement du cerveau humain et les impacts de certaines pathologies sur les connections neuronales, des scientifiques ont modélisés le réseau de connexions neuronales sous forme d'un connectome. Ce rapport concerne le projet de recherche et de développement d'une application permettant de visualiser et comparer des connectomes à l'aide de graphes pour étudier les connexions du cerveau humain.

## Mots-clés

Connectome, Comparaison, Fourmis, Visualisation, Modélisation

## Abstract

To better understand the workings of the human brain and the effects of some diseases on neuronal connections, scientists modeled the network of neuronal connections under the term of connectome. This report is about the research and development project concerning an application to visualize and compare connectomes using graphs to study the connections in human brains.

## Keywords

Connectome, Comparison, Ants, Visualization, Modelling

## Tuteurs académiques

Nicolas MONMARCHÉ  
Kieu Diem Ho  
Jean-Yves RAMEL

## Étudiant

Clément CONDETTE (DI5)