République Algérienne Démocratique et Populaire Ministère de l'Enseignement Supérieur et de la Recherche Scientifique Université des Sciences et de la Technologie Houari Boumediene

Faculté d'informatique - Département d'informatique

Rapport en EOC
Master 1 en Bio-informatique

Thème

Réseaux PPI

Nom et Prénom :

BOUBRIMA Ali

MAMMERI Sara

Chargé de Cours : Mme MEHDI

1. Introduction

Les réseaux d'interaction protéine-protéine (PPI) sont des représentations graphiques des interactions physiques entre les protéines au sein d'une cellule. En biologie, ces réseaux revêtent une grande importance car ils permettent de mieux comprendre les processus cellulaires ainsi que les maladies qui y sont associées.

Le matching de sous-graphes, quant à lui, est une technique largement utilisée pour trouver des motifs isomorphes entre deux graphes. Dans le contexte des réseaux PPI, cette technique est appliquée pour l'alignement de réseaux, ce qui permet d'identifier les protéines similaires présentes dans différents réseaux PPIs.

Ce rapport a pour objectif de fournir une description détaillée des réseaux PPI, des techniques de matching de sous-graphes et des bibliothèques Python disponibles pour leur implémentation.

Nous commencerons par expliquer en quoi consistent les réseaux PPI et comment ils sont construits à partir des données expérimentales ou de prédiction. Nous aborderons également les principaux concepts et mesures utilisés pour analyser ces réseaux, tels que les degrés de connectivité, les clusters et les motifs.

Ensuite, nous nous pencherons sur la technique du matching de sous-graphes, qui permet de détecter les motifs similaires entre deux réseaux PPI. Nous expliquerons les différentes étapes de ce processus, de la recherche des sous-graphes isomorphes à la mesure de similarité entre les motifs identifiés.

Enfin, nous présenterons les bibliothèques Python les plus couramment utilisées pour l'implémentation du matching de sous-graphes dans les réseaux PPI. Nous détaillerons leurs fonctionnalités, leurs avantages et leurs limites, afin de guider les chercheurs dans le choix de l'outil adapté à leurs besoins.

Ce rapport fournira ainsi une base solide pour comprendre et mettre en œuvre le matching de sous-graphes dans le contexte des réseaux PPI, contribuant ainsi à l'avancement de la recherche en biologie et en médecine.

2. Réseaux PPI

Les réseaux d'interaction protéine-protéine (PPI) sont des graphes qui offrent une représentation visuelle des interactions physiques entre les protéines à l'intérieur d'une cellule. Ces réseaux jouent un rôle essentiel en biologie en nous permettant de mieux comprendre les processus cellulaires et les maladies qui y sont liées.

Dans un réseau PPI, chaque protéine est représentée par un nœud, tandis que les interactions entre les protéines sont représentées par des arêtes reliant les nœuds correspondants. Ainsi, chaque arête représente une interaction physique entre deux protéines spécifiques. En étudiant ces interactions, nous pouvons obtenir des informations précieuses sur les voies de signalisation, les régulations génétiques, les processus métaboliques et bien d'autres aspects de la biologie cellulaire.

Les réseaux PPIs peuvent être construits à partir de différentes sources de données. Les approches expérimentales comprennent des techniques telles que les co-immunoprécipitations, les levures à deux hybrides, les spectrométries de masse, etc. Ces expériences permettent de détecter physiquement les interactions entre les protéines et de construire un réseau en conséquence. D'autre part, des méthodes de prédiction basées sur des algorithmes peuvent également être utilisées pour inférer les interactions entre les protéines à partir de données génomiques, protéomiques ou de séquence.

En utilisant les réseaux PPIs, les chercheurs peuvent étudier et analyser divers aspects de la biologie cellulaire. Par exemple, l'identification des sous-réseaux fortement connectés (clusters) peut révéler des complexes protéiques fonctionnels ou des voies de signalisation spécifiques. L'analyse des motifs, qui sont des sous-structures récurrentes dans le réseau, permet de mettre en évidence des schémas d'interaction particuliers. De plus, l'étude des propriétés topologiques du réseau, telles que les degrés de connectivité des protéines, la centralité ou la modularité, peut fournir des informations sur l'importance et le rôle des protéines dans le contexte du réseau PPI global.

En somme, les réseaux PPIs constituent des outils puissants pour comprendre les interactions entre les protéines et leur influence sur les processus biologiques. En combinant des approches expérimentales et informatiques, ces réseaux permettent d'obtenir des informations approfondies sur la structure, la fonction et la régulation des protéines, ainsi que sur les mécanismes sous-jacents aux maladies.

3. Matching de sous-graphes

Le matching de sous-graphes est une méthode essentielle utilisée en analyse de réseaux pour trouver des sous-structures similaires ou isomorphes entre deux graphes. Dans le contexte des réseaux d'interaction protéine-protéine (PPI), cette technique est appliquée à l'alignement de réseaux PPIs afin d'identifier les protéines similaires présentes dans différents réseaux.

L'alignement de réseaux PPI est un processus qui vise à identifier les protéines qui ont des interactions similaires ou conservées entre différentes espèces ou conditions biologiques. Comprendre la conservation de la fonction des protéines est crucial pour établir des liens entre les espèces et pour étudier l'évolution des réseaux d'interaction protéine-protéine. De plus, l'alignement des réseaux PPIs peut également fournir des informations sur les relations entre les maladies, en identifiant les protéines communes impliquées dans des maladies similaires ou des voies de signalisation partagées.

Le processus de matching de sous-graphes dans les réseaux PPIs implique la recherche de motifs ou de sous-structures similaires entre les réseaux. Ces motifs peuvent être des ensembles de protéines interagissant de manière spécifique ou des régions spécifiques du réseau qui présentent des propriétés similaires. La recherche de ces motifs permet d'identifier les protéines similaires entre les réseaux PPIs, ce qui facilite la comparaison et l'analyse fonctionnelle des protéines à travers différentes espèces ou conditions.

La méthode de matching de sous-graphes implique généralement plusieurs étapes. Tout d'abord, il faut sélectionner un motif ou un sous-graphe cible à rechercher dans les réseaux PPIs. Ensuite, des algorithmes de recherche efficaces sont utilisés pour trouver des occurrences de ce motif dans les réseaux. Ces algorithmes peuvent être basés sur des techniques de recherche exhaustive, de programmation dynamique, ou des approches heuristiques pour améliorer l'efficacité de la recherche.

Une fois que des motifs similaires ont été identifiés, une mesure de similarité est généralement utilisée pour quantifier à quel point ces motifs sont semblables. Cette mesure peut être basée sur des critères topologiques, tels que le nombre d'arêtes ou de nœuds communs, ou sur des critères fonctionnels, en utilisant des annotations biologiques ou des informations sur les voies de signalisation.

En somme, le matching de sous-graphes dans les réseaux PPIs est une technique importante pour l'alignement et la comparaison des réseaux d'interaction protéine-protéine. En identifiant les protéines similaires entre différents réseaux, cette méthode permet de mieux comprendre la conservation de la fonction protéique, les relations entre les espèces et les maladies. Elle joue un rôle crucial dans l'élucidation des mécanismes biologiques et la découverte de cibles thérapeutiques potentielles.

4. Algorithmes existants

GraMoFoNe :

Cet algorithme combine l'approche du motif fréquent (frequent pattern mining) avec la recherche de motifs dans les réseaux PPIs. Il utilise des techniques de compression de graphes pour extraire les motifs fréquents, et identifie les sous-graphes similaires dans les réseaux PPIs en se basant sur ces motifs.

SPINAL :

Cet algorithme vise à détecter des structures hiérarchiques (arborescences) dans les réseaux PPIs. Il identifie des ensembles de protéines qui sont fortement connectées les unes aux autres et les considère comme des sous-graphes pertinents. SPINAL utilise des mesures de similarité et des techniques d'optimisation pour trouver les structures hiérarchiques dans les réseaux PPIs.

Algorithme Ullman :

L'algorithme Ullman, développé par John E. Ullman en 1976, est utilisé pour trouver des sous-graphes isomorphes entre deux graphes. Il est basé sur une approche récursive et systématique qui explore toutes les correspondances possibles entre les nœuds des deux graphes.

Voici une description plus détaillée du fonctionnement de l'algorithme Ullman :

Étape d'initialisation : L'algorithme commence par prendre en entrée les deux graphes que l'on souhaite comparer. Il crée deux ensembles de correspondances vides, un pour les nœuds du premier graphe et un pour les nœuds du deuxième graphe.

Étape de correspondance : L'algorithme explore récursivement toutes les correspondances possibles entre les nœuds des deux graphes. Pour chaque nœud du premier graphe, il tente de le faire correspondre à un nœud non apparié du deuxième graphe. Il vérifie ensuite si les arêtes associées à ces nœuds sont également présentes dans les correspondances. Si une correspondance satisfait cette condition, elle est ajoutée aux ensembles de correspondances.

Étape de récursion : Une fois qu'une correspondance est trouvée, l'algorithme passe au nœud suivant dans le premier graphe et continue la recherche de correspondances récursivement. Il répète cette étape jusqu'à ce que tous les nœuds du premier graphe aient été appariés ou jusqu'à ce qu'il n'y ait plus de correspondances possibles.

Rétrogradation (backtracking) : Si une correspondance ne peut pas être trouvée pour un nœud du premier graphe, l'algorithme effectue une rétrogradation et explore d'autres options. Il revient au nœud précédent dans le premier graphe et essaie une autre

correspondance dans le deuxième graphe. Cette étape de rétrogradation permet à l'algorithme d'explorer toutes les possibilités de correspondances.

Terminaison : L'algorithme se termine lorsque toutes les correspondances possibles ont été explorées ou lorsque tous les nœuds du premier graphe ont été appariés. À ce stade, les ensembles de correspondances contiennent les paires de nœuds correspondants entre les deux graphes.

L'algorithme Ullman est considéré comme un algorithme complet, c'est-à-dire qu'il est garanti de trouver toutes les correspondances isomorphes possibles entre les graphes. Cependant, sa complexité est exponentielle par rapport au nombre de nœuds dans les graphes, ce qui peut rendre son exécution lente pour de grands graphes.

Dans le contexte du matching de sous-graphes dans les réseaux PPI, l'algorithme Ullman peut être utilisé pour identifier les protéines similaires entre différents réseaux PPIs. Il permet de trouver des sous-ensembles de protéines qui interagissent de manière similaire dans les différents réseaux, ce qui facilite l'analyse comparative et fonctionnelle des protéines.

Algorithme VF2 :

L'algorithme VF2 (VF2 Algorithm for Graph Isomorphism) est un algorithme utilisé pour trouver des sous-graphes isomorphes entre deux graphes. Il utilise une approche itérative basée sur le principe de recherche incrémentale, en associant progressivement des nœuds correspondants dans les deux graphes.

Voici une description plus détaillée du fonctionnement de l'algorithme VF2 :

Étape d'initialisation : L'algorithme commence par prendre en entrée les deux graphes que l'on souhaite comparer. Il crée une structure de données appelée état VF2, qui contient des informations sur les correspondances actuelles, les nœuds déjà appariés et d'autres paramètres nécessaires à l'algorithme.

Étape de correspondance initiale : L'algorithme commence par essayer de trouver une correspondance initiale entre les nœuds des deux graphes. Pour chaque nœud du premier graphe, il tente de trouver un nœud correspondant dans le deuxième graphe en se basant sur des critères tels que les étiquettes de nœuds et les arêtes adjacentes. Si une correspondance est trouvée, elle est ajoutée à l'état VF2.

Étape de récursion : Une fois qu'une correspondance initiale est établie, l'algorithme passe à une étape de recherche récursive. Il explore récursivement toutes les correspondances possibles entre les nœuds des deux graphes en respectant les contraintes d'isomorphisme.

L'algorithme effectue une série de vérifications pour chaque paire de nœuds correspondants, en s'assurant que les arêtes adjacentes sont également présentes dans les correspondances.

Rétrogradation (backtracking) : Si une correspondance ne peut pas être trouvée pour une paire de nœuds correspondants, l'algorithme effectue une rétrogradation et explore d'autres options. Il revient à l'état précédent de l'état VF2 et tente d'autres correspondances. Cette étape de rétrogradation permet à l'algorithme de continuer à explorer toutes les possibilités de correspondances.

Terminaison : L'algorithme se termine lorsque toutes les correspondances possibles ont été explorées ou lorsque tous les nœuds des graphes ont été appariés. À ce stade, l'état VF2 contient les paires de nœuds correspondants entre les deux graphes.

L'algorithme VF2 est efficace pour trouver des sous-graphes isomorphes dans de nombreux cas, mais sa complexité est exponentielle par rapport au nombre de nœuds dans les graphes. Par conséquent, pour de grands graphes, l'algorithme peut être coûteux en termes de temps de calcul.

Dans le contexte de l'alignement de réseaux PPI, l'algorithme VF2 peut être appliqué pour identifier les protéines similaires entre différents réseaux PPIs. En trouvant des sous-graphes isomorphes, l'algorithme permet de découvrir des motifs de connectivité similaires entre les réseaux, ce qui peut aider à comprendre la conservation de la fonction des protéines et les relations entre les maladies.

L'algorithme de Subgraph Matching

L'algorithme de Subgraph Matching (correspondance de sous-graphes) est une technique utilisée pour trouver des sous-graphes isomorphes ou partiellement isomorphes dans un graphe plus grand. Contrairement aux algorithmes de correspondance de graphes complets, qui cherchent à trouver des correspondances exactes entre les nœuds et les arêtes, l'algorithme de Subgraph Matching permet de trouver des correspondances partielles où seuls certains nœuds et arêtes doivent correspondre.

Voici une description générale de l'algorithme de Subgraph Matching :

Étape d'initialisation : L'algorithme prend en entrée un graphe source (le sous-graphe que l'on cherche à trouver) et un graphe cible (le graphe plus grand dans lequel on recherche le sous-graphe). Il initialise également une structure de données pour stocker les correspondances trouvées.

Étape de recherche : L'algorithme commence la recherche des correspondances en explorant les nœuds du graphe cible. Pour chaque nœud du graphe cible, il tente de trouver

une correspondance dans le graphe source en respectant les contraintes d'isomorphisme. Les contraintes peuvent inclure des correspondances d'étiquettes de nœuds, de degrés de nœuds et de motifs de connectivité.

Validation des correspondances : Une fois qu'une correspondance potentielle est trouvée, l'algorithme procède à la vérification de sa validité en comparant les arêtes adjacentes des nœuds correspondants. Il s'assure que les arêtes correspondantes existent également dans le graphe cible.

Récursion : Si la correspondance est validée, l'algorithme procède à une étape récursive en cherchant des correspondances pour les nœuds adjacents. Il répète ce processus jusqu'à ce que tous les nœuds du graphe source soient appariés ou jusqu'à ce qu'il n'y ait plus de correspondances possibles.

Rétrogradation (backtracking) : Si une correspondance ne peut pas être trouvée pour un nœud ou une arête, l'algorithme effectue une rétrogradation et explore d'autres options. Il revient à un état précédent de la recherche et tente d'autres correspondances pour les nœuds adjacents.

Terminaison : L'algorithme se termine lorsque toutes les correspondances possibles ont été explorées ou lorsque tous les nœuds du graphe source ont été appariés. Les correspondances valides sont stockées dans une structure de données pour être utilisées ultérieurement.

L'algorithme de Subgraph Matching est largement utilisé dans divers domaines, tels que la bioinformatique, l'analyse de réseaux sociaux, la recherche d'information, etc. Il permet d'identifier des motifs et des sous-structures importantes dans les graphes, ce qui peut être utile pour comprendre les relations entre les entités, détecter des anomalies ou des sous-graphes spécifiques.

La complexité de l'algorithme de Subgraph Matching dépend de la taille et de la complexité du graphe cible. Pour de grands graphes, la recherche de correspondances peut être coûteuse en termes de temps de calcul. Par conséquent, des optimisations et des techniques d'indexation sont souvent utilisées

5. Sources de données biologiques :

Il existe diverses bases de données biologiques qui recensent des réseaux PPIs et des annotations fonctionnelles des protéines. Les sources de données les plus utilisées sont :

La base de données STRING (Search Tool for the Retrieval of Interacting Genes/Proteins):

La base de données STRING (Search Tool for the Retrieval of Interacting Genes/Proteins) est une ressource en ligne qui permet d'explorer les interactions entre les gènes et les protéines. Elle est l'une des meilleures. Elle contient 3 123 056 667 interactions au total au moment de la rédaction de cet article. Ces interactions impliquent plus de 20 000 000 de protéines dans plus de 5 000 organismes, se combinant pour se former. STRING permet aux chercheurs de rechercher des gènes ou des protéines spécifiques, d'analyser les interactions associées et d'effectuer des analyses globales des réseaux. C'est un outil essentiel pour la recherche en biologie et en bioinformatique, facilitant la compréhension des mécanismes moléculaires et des relations entre les biomolécules.

DIP (Database of Interacting Proteins) : DIP est une base de données en ligne qui recense les interactions protéine-protéine expérimentalement prouvées. Elle fournit des informations sur les protéines impliquées dans les interactions, les techniques expérimentales utilisées pour les détecter, ainsi que des données sur les complexes protéiques et les voies de signalisation.

GO (Gene Ontology) : GO est une base de données en ligne qui fournit un ensemble de termes ontologiques pour décrire les fonctions biologiques, les processus et les composants cellulaires. Ces termes permettent d'annoter les protéines et de les classifier en fonction de leur rôle dans les processus biologiques.

BioGRID (Biological General Repository for Interaction Datasets):

BioGRID (Biological General Repository for Interaction Datasets) est une base de données en ligne qui stocke et met à disposition des données sur les interactions entre les biomolécules, notamment les protéines. Elle est largement utilisée par la communauté de la recherche en biologie et en bioinformatique pour étudier les interactions protéine-protéine et les réseaux d'interaction.

BioGRID recueille des données expérimentales sur les interactions entre les biomolécules à partir de diverses sources, notamment des publications scientifiques, des bases de données publiques, des expériences de co-immunoprécipitation, des levures à deux hybrides, des spectrométries de masse, des études de protéomique, etc. Ces données sont soigneusement

annotées et intégrées dans BioGRID pour fournir des informations fiables et de haute qualité.

La base de données BioGRID contient des informations sur les interactions protéine-protéine, les interactions protéine-ARN, les interactions protéine-ADN, les interactions protéine-ligand, ainsi que d'autres types d'interactions biomoléculaires. Les données sont organisées sous forme de jeux de données spécifiques, permettant aux chercheurs d'accéder facilement aux informations pertinentes pour leurs études.

BioGRID fournit une interface conviviale permettant aux utilisateurs d'interroger la base de données et d'effectuer des recherches spécifiques. Les utilisateurs peuvent effectuer des recherches en fonction de critères tels que les noms de protéines, les identifiants d'interaction, les organismes d'origine, les techniques expérimentales utilisées, etc. Les résultats de recherche sont présentés de manière claire et peuvent être téléchargés pour une analyse plus poussée.

En plus de fournir des données brutes sur les interactions biomoléculaires, BioGRID offre également des outils d'analyse et de visualisation pour aider les chercheurs à explorer et à interpréter les réseaux d'interaction. Ces outils permettent de générer des graphiques interactifs, de calculer des mesures de centralité, de détecter des communautés et d'effectuer d'autres analyses de réseaux.

En résumé, BioGRID est une ressource essentielle pour la communauté de recherche en biologie et en bioinformatique, offrant des données précieuses sur les interactions biomoléculaires. Elle facilite la compréhension des réseaux d'interaction et contribue à l'avancement des connaissances dans des domaines tels que la biologie des systèmes, la biologie des réseaux et la découverte de médicaments.

La base de données IntAct (Integrated Interaction Database) :

Est une ressource en ligne qui recueille et met à disposition des informations sur les interactions entre les protéines. Elle fournit des détails sur les preuves expérimentales, les méthodes d'identification et les participants des interactions protéine-protéine. IntAct permet aux chercheurs d'explorer ces interactions, de rechercher des protéines spécifiques et d'accéder aux informations détaillées des expériences. Elle offre également des outils d'analyse et de visualisation pour faciliter la compréhension des réseaux d'interaction protéine-protéine. IntAct est une ressource précieuse pour la recherche en biologie et en bioinformatique, offrant un accès aux connaissances sur les interactions protéine-protéine.

6. Bibliothèques Python :

NetworkX : NetworkX est une bibliothèque Python largement utilisée pour l'analyse de réseaux. Elle offre des outils puissants pour la création, la manipulation et l'analyse de graphes. NetworkX fournit des fonctionnalités pour le calcul de la similarité entre les graphes, la recherche de sous-graphes isomorphes et l'alignement de réseaux PPI.

igraph : igraph est une autre bibliothèque Python populaire pour l'analyse de graphes. Elle propose des fonctionnalités similaires à NetworkX, permettant la création, la manipulation et l'analyse de graphes. igraph offre des outils avancés pour l'exploration des propriétés topologiques des graphes et l'analyse des communautés.

Graph-tool : une bibliothèque Python pour la manipulation et l'analyse de graphes, qui contient des fonctions pour l'alignement de graphes, y compris l'algorithme d'Ullman.

PyMatching : une bibliothèque Python pour l'alignement de graphes, qui implémente plusieurs algorithmes de matching de sous-graphes, y compris VF2 et Ullman.

Exemple d'implémentation :

Voici un exemple simplifié d'implémentation de l'alignement de réseaux PPI en utilisant la bibliothèque NetworkX :

- Télécharger les données des réseaux PPIs à partir de DIP ou GO, en récupérant les interactions protéine-protéine et les annotations biologiques.
- Charger les données des réseaux PPIs dans des objets graphiques NetworkX en créant des nœuds pour les protéines et des arêtes pour les interactions.
- Utiliser des mesures de similarité, telles que le coefficient de Jaccard ou le coefficient de similarité cosinus, pour calculer la similarité entre les réseaux PPIs.
- Appliquer l'algorithme VF2 de NetworkX pour trouver des sous-graph

Nous allons l'implémenter avec Python dans la partie suivante.

7. Programme python

[Le même code est joint en tant que fichier de notebook Python pour une meilleure compréhension.]

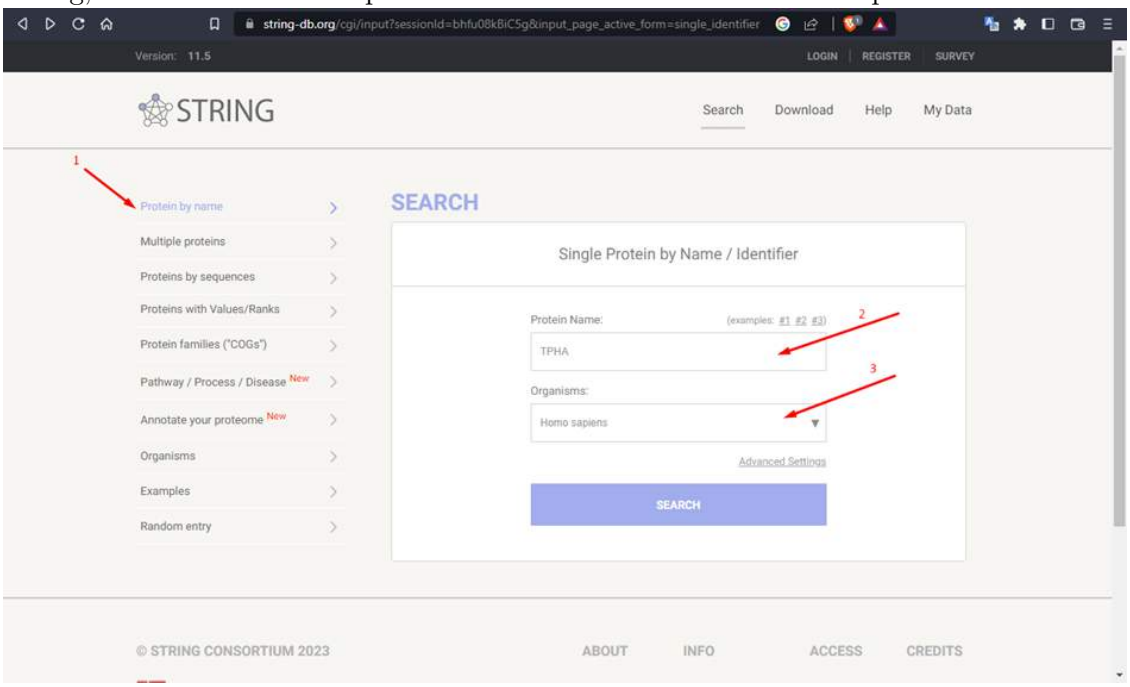
ppi

June 1, 2023

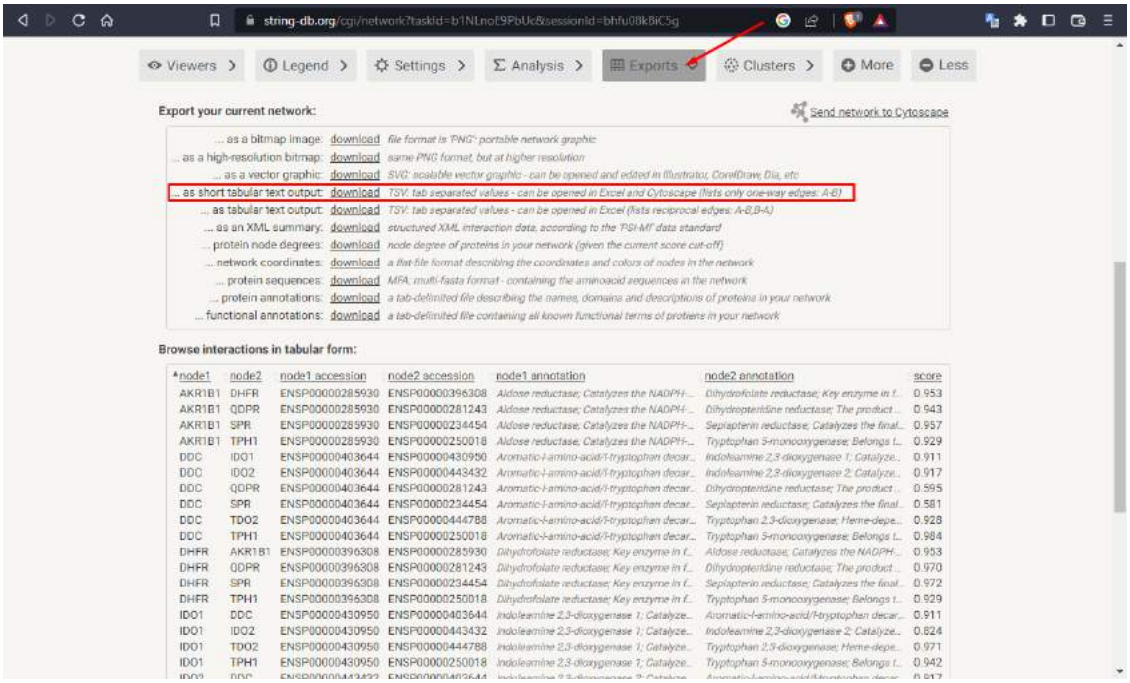
[1]:
```
# Importation des bibliothèques
import networkx as nx
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from networkx.algorithms import isomorphism
from IPython.display import Image
```

1 Collecte et Préparation des Données

Tout d'abord, nous nous rendons sur le site web https://string-db.org, où nous spécifions le nom de la protéine et l'organisme



Nous cliquons sur "export" et téléchargeons notre dataset.



Essentiellement, ce fichier TSV exporté à partir du site web STRING-DB contient des informations sur les interactions protéine-protéine (PPI) impliquant la protéine sélectionnée (TPH1). Le fichier est au format de valeurs séparées par des tabulations, ce qui permet de l'ouvrir dans Excel ou Cytoscape, ou simplement comme un fichier texte.

1.1 Lecture des données du fichier tsv

[2]:
```
# Convertir en dataframe en utilisant la première ligne comme noms de colonnes ;
  ↪ supprimer la dernière ligne vide
df = pd.read_csv('./dataset.tsv', delimiter='\t', skipfooter=1, engine='python')
df.head()
```

[2]:

	#node1	node2	node1_string_id	node2_string_id	\
0	AKR1B1	SPR	9606.ENSP00000285930	9606.ENSP00000234454	
1	AKR1B1	TPH1	9606.ENSP00000285930	9606.ENSP00000250018	
2	AKR1B1	FDX1	9606.ENSP00000285930	9606.ENSP00000260270	
3	AKR1B1	SORD	9606.ENSP00000285930	9606.ENSP00000267814	
4	AKR1B1	QDPR	9606.ENSP00000285930	9606.ENSP00000281243	

	neighborhood_on_chromosome	gene_fusion	phylogenetic_cooccurrence	\
0	0.041	0.0	0.0	
1	0.000	0.0	0.0	
2	0.000	0.0	0.0	
3	0.085	0.0	0.0	
4	0.041	0.0	0.0	

	homology	coexpression	experimentally_determined_interaction	\
0	0.0	0.062	0.104	

```
1       0.0         0.062                                   0.000
2       0.0         0.057                                   0.000
3       0.0         0.099                                   0.494
4       0.0         0.062                                   0.248

   database_annotated  automated_textmining  combined_score
0                 0.9                 0.550           0.957
1                 0.9                 0.307           0.929
2                 0.0                 0.987           0.987
3                 0.9                 0.905           0.995
4                 0.9                 0.295           0.943
```

Il y a plusieurs attributs (colonnes) pour chaque interaction, mais nous nous concentrerons uniquement sur 3 d'entre eux :

- #node1 : L'identifiant ou le nom de la première protéine dans l'interaction (A dans l'arête unidirectionnelle A-B).
- node2 : L'identifiant ou le nom de la deuxième protéine dans l'interaction (B dans l'arête unidirectionnelle A-B).
- combined_score : Un score ou une mesure de confiance indiquant la force ou la fiabilité de l'interaction entre les deux protéines. Des scores plus élevés indiquent généralement des interactions plus fiables.

[3]:
```
# Dataframe avec les noms préférés des deux protéines et le score de
↪l'interaction
interactions = df[['#node1', 'node2', 'combined_score']]
interactions.head()
```

[3]:
```
   #node1 node2  combined_score
0  AKR1B1   SPR           0.957
1  AKR1B1  TPH1           0.929
2  AKR1B1  FDX1           0.987
3  AKR1B1  SORD           0.995
4  AKR1B1  QDPR           0.943
```

1.2 Création d'un graphe

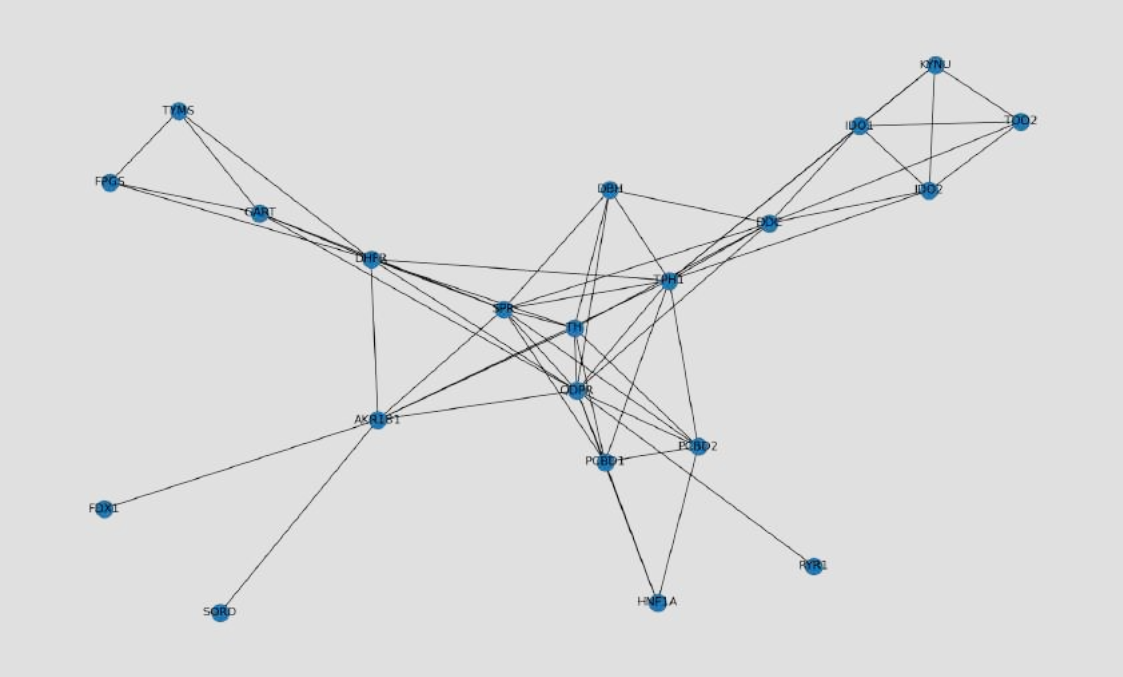
Le code suivant montre comment ces données peuvent être utilisées pour construire le graphe avec NetworkX. Chaque nœud représente une protéine, chaque arête représente une interaction entre deux protéines, et chaque arête est pondérée par le score.

[4]:
```
G = nx.Graph(name="Graphe d interaction des protéines")

interactions = np.array(interactions)
# Ajout des arêtes pondérées au graph
for i in range(len(interactions)):
    interaction = interactions[i]
    a = interaction[0]  # noeud protéine a
```

```
    b = interaction[1]  # noeud protéine b
    w = float(interaction[2])  # score en tant qu'arête pondérée où les scores␣
 ↪élevés = faible poids
    G.add_weighted_edges_from([(a, b, w)])  # ajouter une arête pondérée au␣
 ↪graphe

num_nodes = G.number_of_nodes()
num_edges = G.number_of_edges()
print("Nombre de noeuds :", num_nodes)
print("Nombre d'arêtes :", num_edges)
```

```
Nombre de noeuds : 21
Nombre d'arêtes : 58
```

[5]:
```
# Affichage du graphe
# nx.spring_layout is one of NetworkX's node positioning algorithms.
pos = nx.spring_layout(G)  # positionner les noeuds en utilisant le placement␣
 ↪en ressort
plt.figure(figsize=(20, 12), facecolor=[0.7, 0.7, 0.7, 0.4])
nx.draw_networkx(G)
plt.axis('off')
plt.show()
```

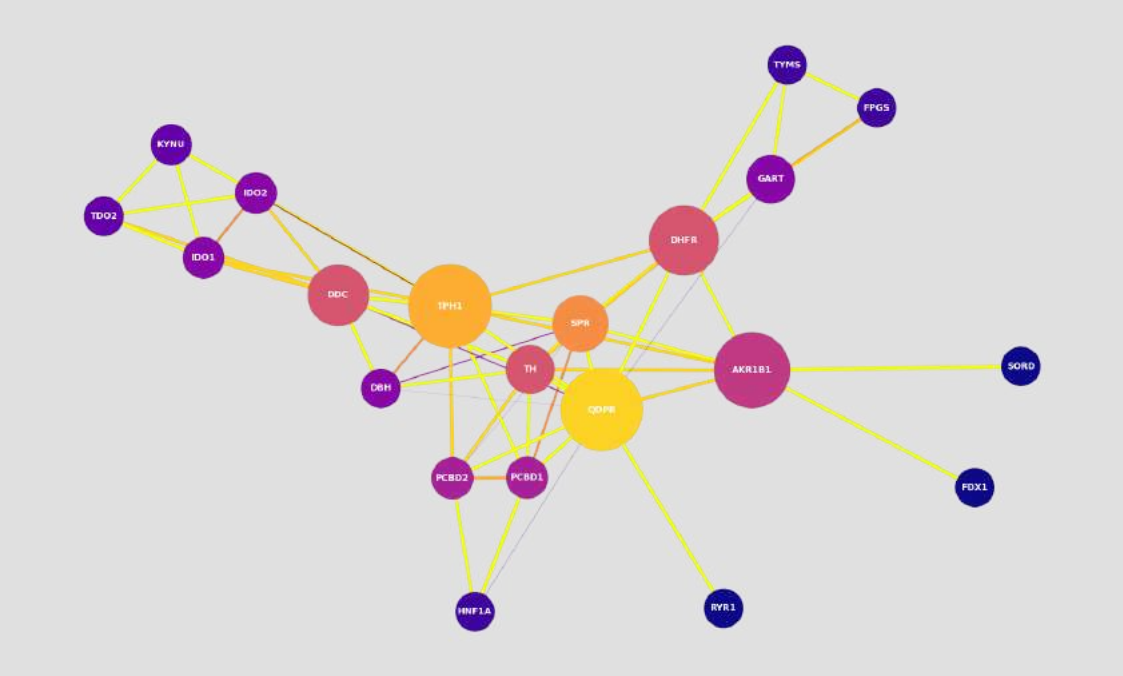


La visualisation simple montrée ci-dessus ne nous fournit pas beaucoup d'aide, c'est pourquoi nous essayons de personnaliser notre graphe en écrivant le code ci-dessous qui nous aidera de la manière suivante :

La gamme de couleurs va du violet foncé au jaune vif. Plus le nœud est jaune, plus son degré est élevé. Plus le nœud est grand, plus sa centralité d'intermédiarité est élevée. Plus l'arête est jaune et large, plus le score d'interaction (poid) est élevé.

[6]:
```
# Fonction pour mettre à l'échelle une liste de valeurs dans une plage [newmin,␣
 ↪newmax]
def rescale(l, newmin, newmax):
    arr = list(l)
    return [(x - min(arr)) / (max(arr) - min(arr)) * (newmax - newmin) + newmin␣
 ↪for x in arr]

# Utiliser la colormap plasma de matplotlib
graph_colormap = cm.get_cmap('plasma', 12)
# La couleur des noeuds varie avec le degré
c = rescale([G.degree(v) for v in G], 0.0, 0.9)
c = [graph_colormap(i) for i in c]
# La taille des noeuds varie avec la centralité de l'intermédiarité - mise à␣
 ↪l'échelle entre [10,100]
bc = nx.betweenness_centrality(G)  # centralité de l'intermédiarité
s = rescale([v for v in bc.values()], 1500, 7000)
# L'épaisseur des arêtes montre 1-poids pour convertir le coût en force de␣
 ↪l'interaction
ew = rescale([float(G[u][v]['weight']) for u, v in G.edges], 0.1, 4)
# La couleur des arêtes montre également le poids
ec = rescale([float(G[u][v]['weight']) for u, v in G.edges], 0.1, 1)
ec = [graph_colormap(i) for i in ec]
```

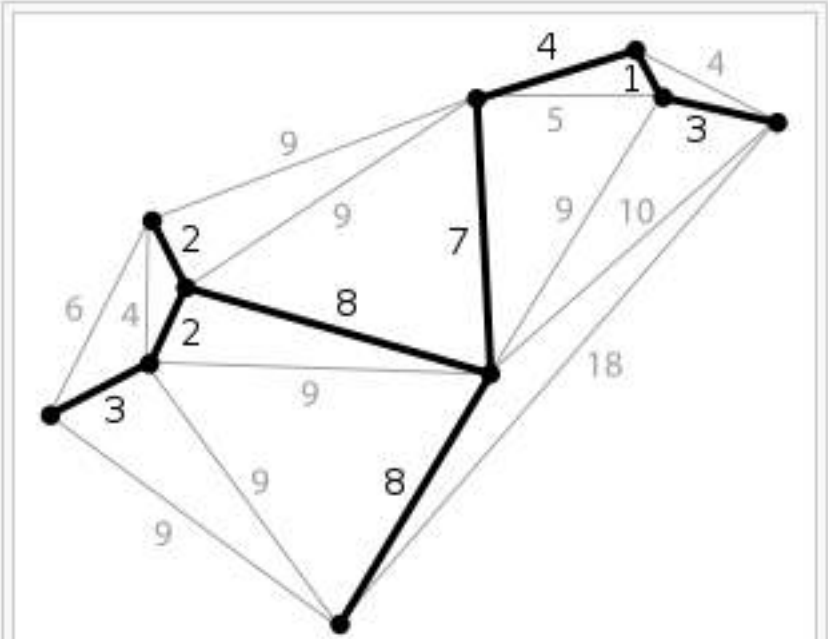
[7]:
```
pos = nx.spring_layout(G)
plt.figure(figsize=(20, 12), facecolor=[0.7, 0.7, 0.7, 0.4])
nx.draw_networkx(G, pos=pos, with_labels=True, node_color=c, node_size=s,␣
 ↪edge_color=ec, width=ew,
                 font_color='white', font_weight='bold', font_size='9')
plt.axis('off')
plt.show()
```



Rendons cela encore plus intéressant en essayant d'afficher l'arbre couvrant de poids minimum ("minimum spanning tree") :

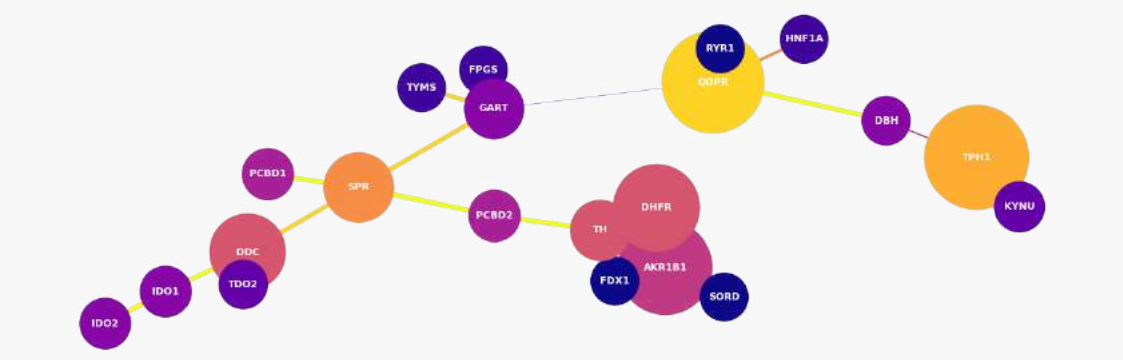
Les arbres couvrants de poids minimum sont largement utilisés dans les réseaux de protéines-proteines (PPI) pour plusieurs applications. Par exemple :

Pour la prédiction de nouvelles interactions : En identifiant les arêtes de poids minimum manquantes dans l'arbre couvrant, on peut obtenir des indications sur de potentielles interactions non découvertes ou non documentées.



A planar graph and its minimum spanning tree. Each edge is labeled with its weight, which here is roughly proportional to its length.

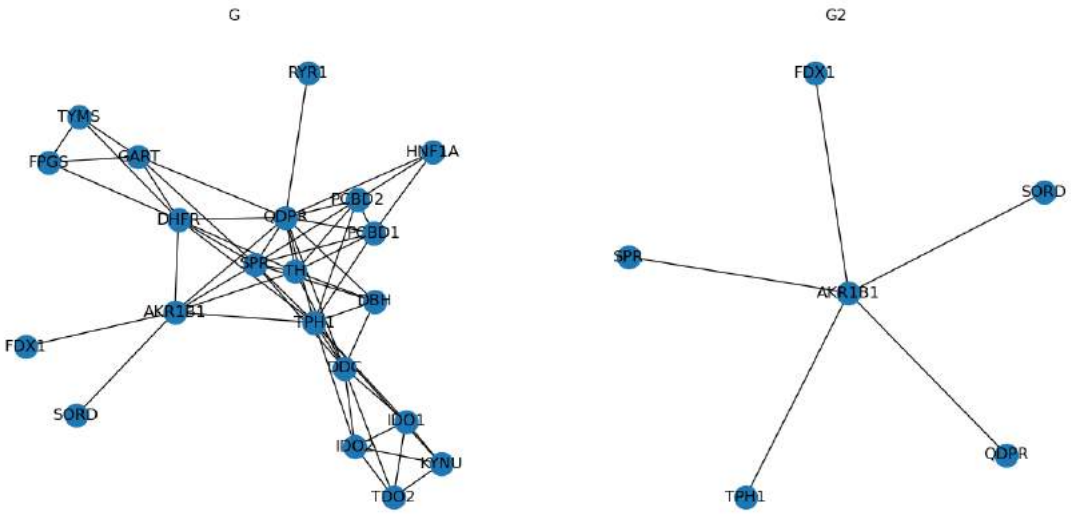
[8]:
```
T = nx.minimum_spanning_tree(G)
pos = nx.spring_layout(T)
plt.figure(figsize=(16, 5), facecolor=[0.7, 0.7, 0.7, 0.1])
nx.draw_networkx(T, pos=pos, with_labels=True, node_color=c, node_size=s,␣
↪edge_color=ec, width=ew,
                 font_color='white', font_weight='bold', font_size='8')
plt.axis('off')
plt.show()
```



Finalement, essayons de réaliser l'isomorphisme entre deux graphes (nous utilisons une autre dataset

des protéines SPR) :

[9]:
```
df2 = pd.read_csv('./dataset1.tsv', delimiter='\t', skipfooter=1,␣
 ↪engine='python')
interactions2 = df2[['#node1', 'node2', 'combined_score']]
G2 = nx.Graph(name="Graphe d'interaction des protéines 2")

interactions2 = np.array(interactions2)
for i in range(len(interactions2)):
    interaction = interactions2[i]
    a = interaction[0]  # noeud protéine a
    b = interaction[1]  # noeud protéine b
    w = float(interaction[2])  # score en tant qu'arête pondérée où les scores␣
 ↪élevés = faible poids
    G2.add_weighted_edges_from([(a, b, w)])  # ajouter une arête pondérée au␣
 ↪graphe

# Afficher les graphes initiaux
plt.figure(figsize=(15, 7))
plt.subplot(121)
nx.draw(G, with_labels=True)
plt.title("G")
plt.subplot(122)
nx.draw(G2, with_labels=True)
plt.title("G2")
plt.show()

# Créer l'objet de comparaison de graphes
gm = isomorphism.GraphMatcher(G, G2)

# Vérifier si G2 est isomorphe à un sous-graphe de G
is_isomorphic = False
for subgraph in gm.subgraph_isomorphisms_iter():
    if len(subgraph) == len(G2):
        print(subgraph)
        is_isomorphic = True
        break

# Afficher le résultat
if is_isomorphic:
    print("G2 est isomorphe à un sous-graphe de G.")
else:
    print("G2 n'est pas isomorphe à aucun sous-graphe de G.")
```




```
{'QDPR': 'AKR1B1', 'DDC': 'SPR', 'RYR1': 'TPH1', 'HNF1A': 'FDX1', 'GART':
'SORD', 'AKR1B1': 'QDPR'}
G2 est isomorphe à un sous-graphe de G.
```

1.3 Autres fonctions

[10]:
```
nx.is_eulerian(G2)
```

[10]:
```
False
```

[11]:
```
nx.vf2pp_is_isomorphic(G, G2)
```

[11]:
```
False
```

[12]:
```
nx.maximal_matching(G)
```

[12]:
```
{('AKR1B1', 'SPR'),
 ('HNF1A', 'PCBD2'),
 ('IDO1', 'KYNU'),
 ('IDO2', 'TDO2'),
 ('QDPR', 'DDC'),
 ('TH', 'DHFR'),
 ('TPH1', 'DBH'),
 ('TYMS', 'FPGS')}
```

8. Conclusion

En conclusion, le matching de sous-graphes est une méthode précieuse pour l'analyse des réseaux biologiques, en particulier les réseaux d'interaction protéine-protéine. Les algorithmes tels que Ullmann et VF2 permettent de trouver des motifs communs dans ces réseaux, ce qui ouvre des opportunités de découvrir des voies de signalisation clés et d'approfondir notre compréhension des processus biologiques.

Les bases de données telles que DIP, STRING, BioGRID et GO jouent un rôle essentiel en fournissant des informations sur les interactions protéine-protéine et gène-protéine, permettant ainsi d'enrichir les analyses de matching de sous-graphes. Elles constituent des ressources précieuses pour les chercheurs qui souhaitent explorer les interactions complexes au sein des réseaux biologiques.

Grâce à l'utilisation des bibliothèques Python comme NetworkX, les chercheurs disposent d'outils puissants pour importer, analyser et visualiser les données des réseaux biologiques. Cela facilite la réalisation d'analyses de sous-graphes et de motifs, aidant ainsi à identifier des structures récurrentes et des protéines clés qui peuvent jouer un rôle central dans les réseaux biologiques.

En résumé, la recherche dans le domaine de la visualisation des données a montré à plusieurs reprises que les êtres humains sont particulièrement doués pour trouver des motifs dans les données visuelles. Il faut moins de temps et moins d'efforts cognitifs pour comprendre un graphique que pour recueillir les mêmes informations à partir d'un tableau. Les visualisations comme celles décrites ici sont d'excellents outils pour examiner et présenter tous types de données de réseau.