Graph Compression Documentation

Arianna Rigamonti - 252321 Bioinformatics for Computational Genomics 2024 - Scientific Programming course

Project 7: Graph Compression - Programming language: Python (v. 3.12.0)

Overview

Graph Compression is a Python script designed to condense protein interaction networks using Spectral Clustering. This script allows users to provide protein information (name and species) and utilizes the STRING API to retrieve protein interaction data. Alternatively, users can directly input their existing interaction dataset.csv. The script then creates an interaction graph and applies spectral clustering to group nodes based on their connectivity patterns. The result is a new, condensed network, which is visualized and saved as both a graph image and a CSV file, facilitating the analysis of the simplified interaction network.

Features

- Retrieve protein interaction data either from the STRING database API or from a user-provided dataset.
- Create a graph representation of the protein interaction network.
- Apply spectral clustering to group nodes in the network.
- Compress the network based on the clustering results.
- Visualize and save the compressed network graph.
- Save the compressed interaction network data as a CSV file.
- Option to overwrite existing files or create unique filenames to prevent overwriting.

Instructions before running the script

Before running the script, ensure that the file `graph_compression.py` is executable. If not, you can make it executable using the following command in the terminal:

```
chmod +x "./graph_compression.py"
```

Dependencies

The script requires the following Python libraries:

- **argparse**: Provides a mechanism for parsing command-line arguments in Python.
- **requests**: Python HTTP library that allows user to send HTTP requests easily.
- **os**: Provides a way of using operating system-dependent functionality. It allows user to interact with the operating system by providing functions for accessing files and directories, working with processes, and more.
- **pandas**: Data manipulation library for DataFrame operations.
- **io**: Provides classes for handling streams of data.
- **networkx**: Library for creating and manipulating complex networks. It provides tools for generating graphs, adding nodes and edges, computing network properties, and visualizing graphs.
- **matplotlib**: Plotting library for visualizing graphs and data.
- **scikit-learn**: Machine learning library with tools for data mining and data analysis. In this script used for performing Spectral Clustering.

To install dependencies, run:

```
pip install requests pandas networkx matplotlib scikit-learn
```

Usage

To run the script, use the following command:

```
python graph_compression.py -p PROTEIN_NAME -s SPECIES -n NODES_NUMBER -c CLUSTERS_NUMBER -o OVERWRITE
```

or

```
python graph_compression.py -d DATA_FRAME -n NODES_NUMBER -c CLUSTERS_NUMBER -o OVERWRITE
```

where: - `PROTEIN_NAME` is the name or identifier of the protein. - `SPECIES` is the taxonomy identifier of the species (default is 9606 for humans). - `NODES_NUMBER` is the number of interacting partners to add. - `CLUSTERS_NUMBER` is the number of clusters for spectral clustering. - `DATA_FRAME` is the path to the input CSV file. - `OVERWRITE` is a boolean flag to specify whether to overwrite existing files.

Example

```
python graph_compression.py -p TP53 -s 9606 -n 20 -c 10 -o True
```

or

```
python graph_compression.py -d /my_path/TP53_interaction_network.csv -n 20 -c 10 -o True
# TP53_interaction_network.csv is available as test dataset in example_output/output_files/interaction_network
```

Classes and Methods

Protein

`__init__(self, overwrite name, species, nodes_number)`

Initializes a Protein object with the following parameters: - `overwrite` : Boolean flag to specify whether to overwrite existing files. - `name` : Name or identifier of the protein. - `species` : Taxonomic identifier of the species (e.g., 9606 for humans). - `nodes_number` : Number of interacting partners to add.

`Protein.get_identifier()`

- **Description:** Retrieves STRING identifier associated with the TP53 protein using the STRING API.
- **Parameters:**
  - `name` : Name of the protein (e.g. TP53)
  - `species` : NCBI taxon identifier for the species (e.g. Human is 9606)
- **Returns:** STRING identifier corresponding to the TP53 protein.

Example:

```
>>> Protein.get_identifier("TP53", 9606)
'9606.ENSP00000493482'
```

`Network.get_interaction_network()`

- **Description:** Retrieves protein interaction network data from the STRING API.
- **Parameters:**
  - `identifiers` : self.identifier
  - `add_nodes` : self.nodes_number
- **Returns:** A DataFrame containing the interaction network data.

Example:

```
>>> Network.get_interaction_network()
stringId_A,stringId_B,preferredName_A,preferredName_B,ncbiTaxonId,score,nscore,fscore,pscore,ascore,escore,dscore
,tscore
9606.ENSP00000212015,9606.ENSP00000254719,SIRT1,RPA1,9606,0.412,0,0.0,0.0,0.089,0.317,0.0,0.13
9606.ENSP00000212015,9606.ENSP00000365230,SIRT1,BCL2L1,9606,0.548,0,0.0,0.0,0.051,0.0,0.0,0.544
9606.ENSP00000212015,9606.ENSP00000371475,SIRT1,TP53BP1,9606,0.565,0,0.0,0.0,0.0,0.291,0.0,0.412
...
```

DataFrame

`__init__(self, input_df)`

Initializes a DataFrame object with an input CSV file. Note: dataset must be formatted like STRING interaction network datasets.

`get_input_df_name(self)`

Returns the name of the input DataFrame file.

Example:

```
>>> df = DataFrame("/my_path/input_dataframe.csv")
>>> df.get_input_df_name()
'input_dataframe.csv'
```

Network

`__init__(self, overwrite, cluster_number, protein=None, dataframe=None)`

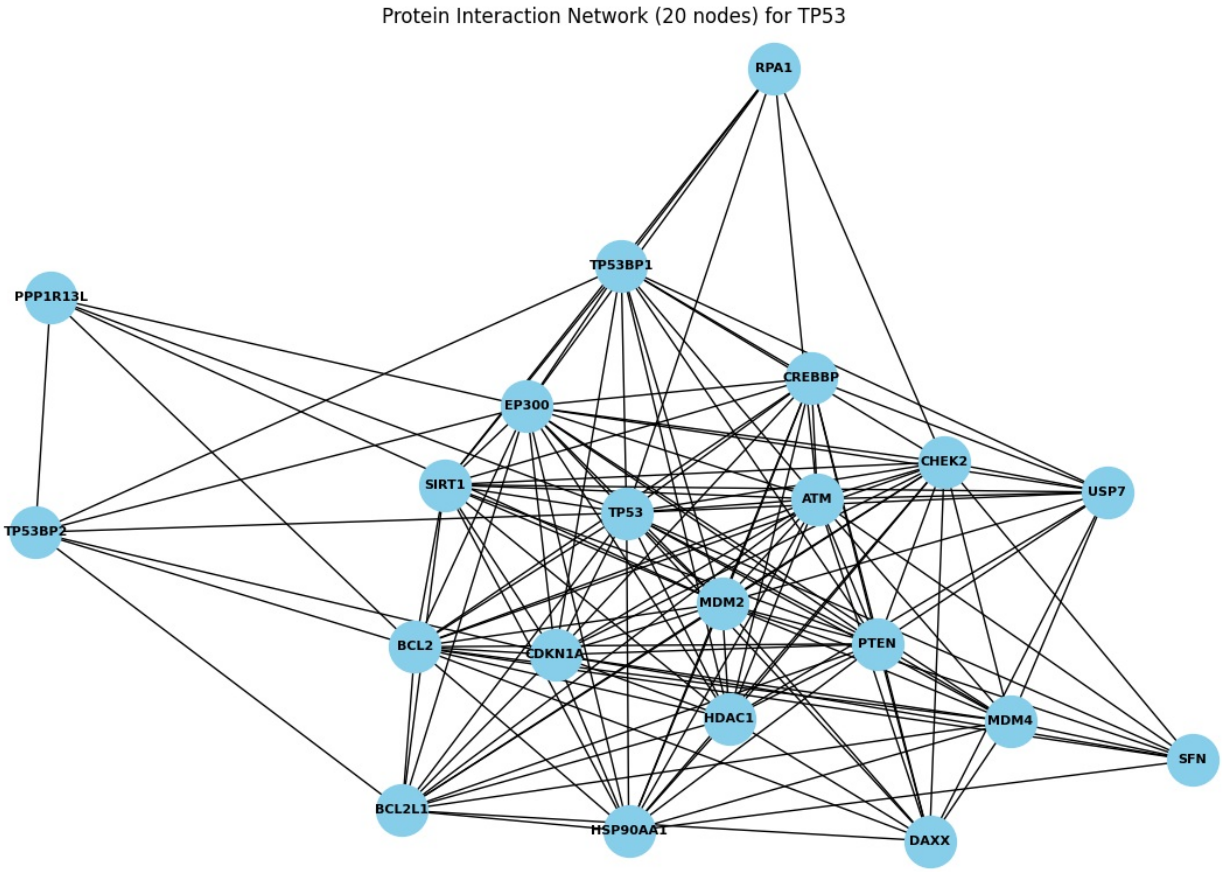
Initializes a Network object with the following parameters: - `overwrite` : Boolean flag to specify whether to overwrite existing files. - `cluster_number` : Number of clusters for spectral clustering. - `protein` : Protein object (optional). - `dataframe` : DataFrame object (optional).

`create_graph(self)`

- **Description:** Creates a graph representation of the protein interaction network.
- **Returns:** A NetworkX graph object representing the protein interaction network.

Example:

```
>>> Network.create_graph()
<networkx.classes.graph.Graph object at 0x7fdba4b0e6a0>
```



Original_graph

`cluster_nodes(self, num_clusters)`

- **Description:** Applies spectral clustering to the graph nodes.
- **Parameters:**
  - `num_clusters` : The number of clusters to create.
- **Returns:** A dictionary mapping nodes to cluster labels.

Example:

```
>>> Network.cluster_nodes(10)
{'ENSP1': 0, 'ENSP2': 1, 'ENSP3': 0, 'ENSP4': 2 ...}
```

`contract_edges(self, node_cluster_mapping)`

- **Description:** Contracts edges in the graph based on the clustering results.
- **Parameters:**
  - `node_cluster_mapping` : A dictionary mapping nodes to cluster labels.
- **Returns:** A compressed NetworkX graph object.

Example:

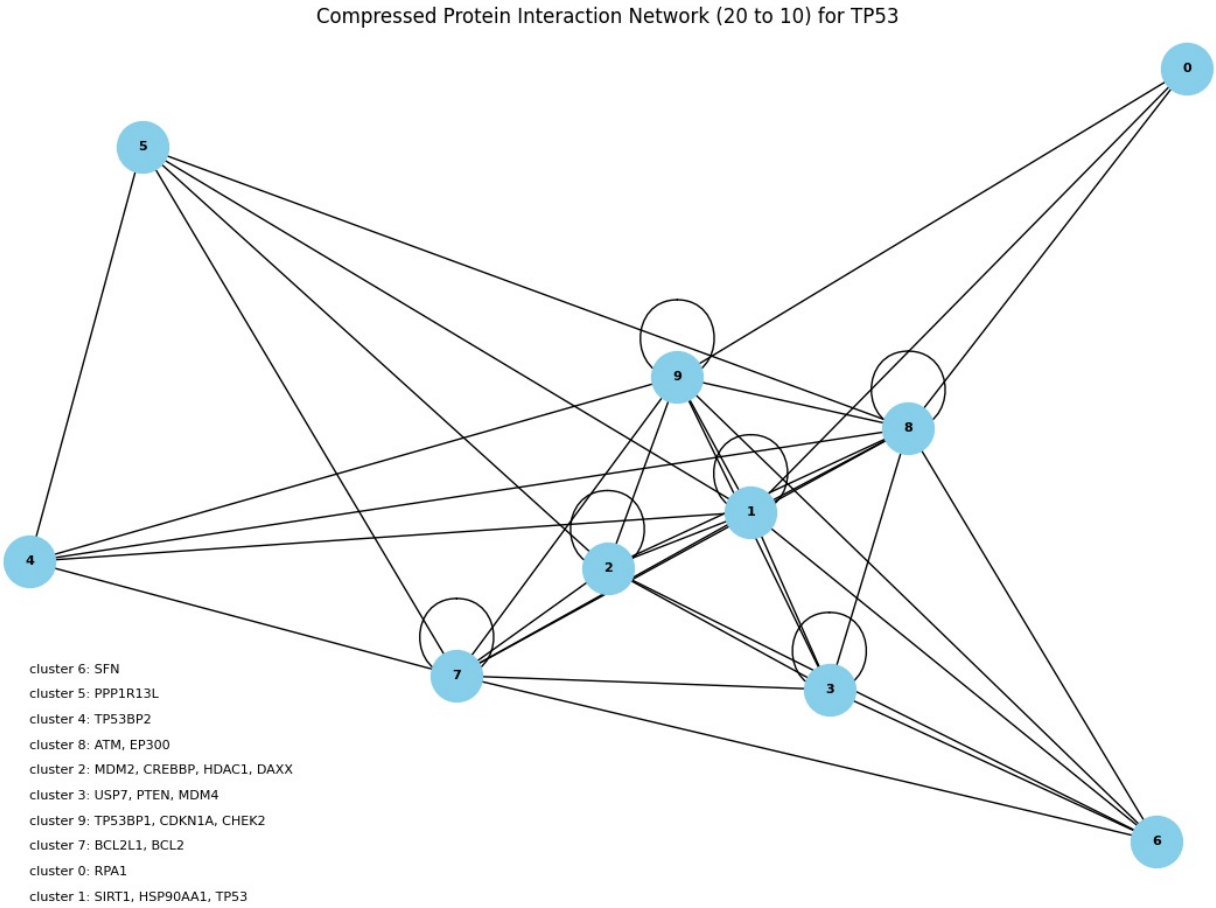
```
>>> Network.contract_edges(node_cluster_mapping)
<networkx.classes.graph.Graph object at 0x7fdba4b0e6a0>
```

`compress_graph(self)`

- **Description:** Compresses the protein interaction network based on spectral clustering. Saves the mapping of original nodes to clusters and the compressed graph as PNG images.
- **Parameters:** None.
- **Returns:** A compressed NetworkX graph object and a dictionary mapping compressed nodes to lists of original nodes.

Example:

```
>>> Network.compress_graph()
(<networkx.classes.graph.Graph object at 0x7fdba4b0e6a0>,
{0: ['ENSP1', 'ENSP3'], 1: ['ENSP2'], 2: ['ENSP4']})
```



Compressed_graph

`get_compressed_interaction_network(self)`

- **Description:** Computes the compressed interaction network data.
- **Returns:** A DataFrame containing the compressed interaction network data. Saves the compressed interaction network to a CSV file.

Example:

```
>>> Network.get_compressed_interaction_network()
clusterA,clusterB,preferred_nameA,preferred_nameB,score
7,3,"SIRT1, HSP90AA1, TP53",RPA1,1.411
7,5,"SIRT1, HSP90AA1, TP53","BCL2L1, BCL2, ATM, EP300",10.366
7,4,"SIRT1, HSP90AA1, TP53","TP53BP1, CDKN1A",4.191
...
```

Functions

`create_directory(path)`

Checks if a directory exists, and if not, creates it.

`get_unique_filepath(filepath)`

Generates a unique filepath by appending a suffix if the file already exists.

`parse_args()`

Parses command-line arguments. Ensures that either a protein name or a data frame is provided, but not both simultaneously.

`main()`

Main function that processes command-line arguments and initializes the appropriate Protein or DataFrame object and the Network object.

Generated Outputs

- **Interaction Network CSV**: The script saves the interaction network data as a CSV file.
- **Original Graph PNG**: The script saves the original interaction network graph as a PNG image.
- **Compressed Graph PNG**: The script saves the compressed interaction network graph as a PNG image.
- **Node Mapping CSV**: The script saves the mapping of original nodes to clusters as a CSV file.
- **Compressed Interaction Network CSV**: The script saves the compressed interaction network data as a CSV file.

Conlusion

This script offers a comprehensive solution for compressing and analyzing protein interaction networks. By leveraging the STRING API and spectral clustering, users can gain insights into the simplified structure of complex interaction networks. The script is flexible, allowing for data input either via protein names or existing CSV datasets, and ensures that all results are saved in a well-organized manner.

For any issues or contributions, please refer to the author.