RINQ_6A5J

May 19, 2025

[1]:
```
!pip install biopython
```

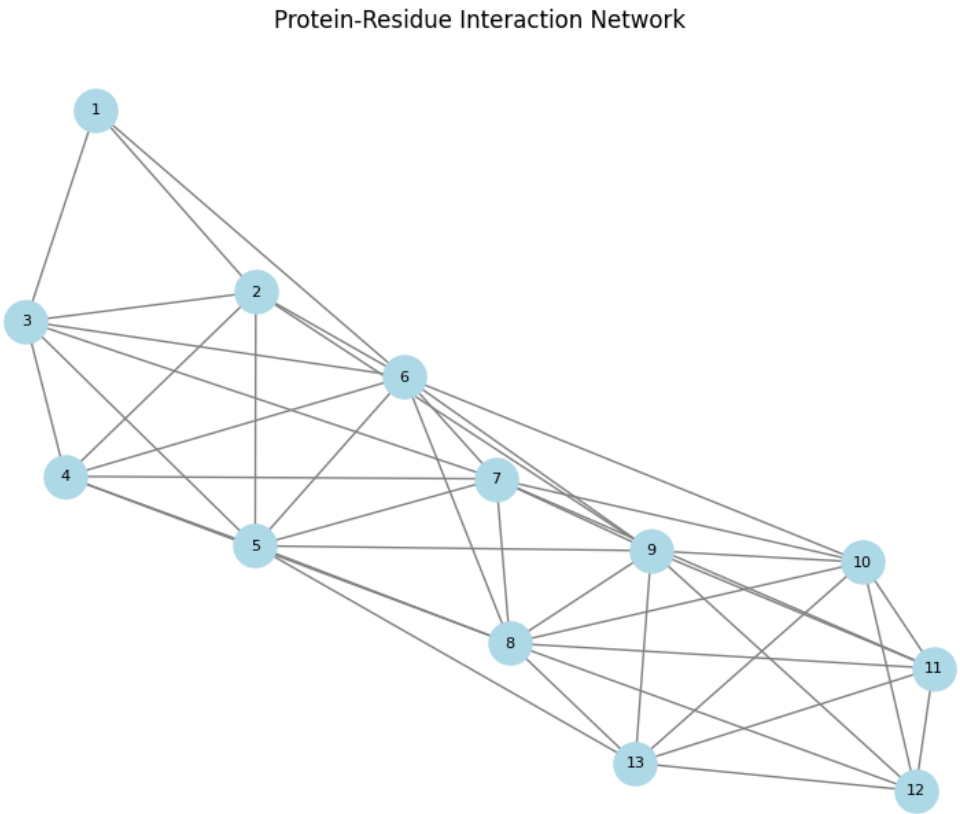
```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: biopython in ./.local/lib/python3.11/site-
packages (1.85)
Requirement already satisfied: numpy in ./.local/lib/python3.11/site-packages
(from biopython) (1.26.4)
DEPRECATION: qiskit-nature 0.5.0 has a non-standard dependency specifier
qiskit-terra>=0.22.*. pip 24.0 will enforce this behaviour change. A possible
replacement is to upgrade to a newer version of qiskit-nature or contact the
author to suggest that they release a version with a conforming dependency
specifiers. Discussion can be found at
https://github.com/pypa/pip/issues/12063
```

[2]:
```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
from Bio import PDB
from Bio.PDB import PDBList

# Function to download PDB file from Protein Data Bank
def fetch_pdb(pdb_id):
    """Fetches a PDB file from the Protein Data Bank given a PDB ID."""
    pdbl = PDBList()
    pdb_file = pdbl.retrieve_pdb_file(pdb_id, file_format='pdb')
    return pdb_file

# Function to calculate the distance between two residues
def calculate_distance(residue_i, residue_j):
    """Calculates the Euclidean distance between the C-alpha atoms of two␣
 ↪residues."""
    try:
        atom_i = residue_i["CA"].coord
        atom_j = residue_j["CA"].coord
```

```
        return np.linalg.norm(atom_i - atom_j)
    except KeyError:
        return float('inf')

# Function to construct a Protein-Residue Interaction Network
def construct_protein_residue_network(pdb_id, interaction_cutoff=8.0):
    """Constructs a residue interaction network from an online PDB file."""
    pdb_path = fetch_pdb(pdb_id)
    parser = PDB.PDBParser()
    structure = parser.get_structure("protein", pdb_path)
    G = nx.Graph()

    for chain in structure.get_chains():
        for residue_i in chain:
            for residue_j in chain:
                if residue_i != residue_j:
                    distance = calculate_distance(residue_i, residue_j)
                    if distance < interaction_cutoff:
                        G.add_edge(residue_i.get_id()[1], residue_j.get_id()[1])

    return G

# Example usage
pdb_id = "6A5J"  # Example PDB ID
interaction_cutoff = 8.0  # Distance cutoff in Angstroms
G = construct_protein_residue_network(pdb_id, interaction_cutoff)

# Extract residue indices from the graph
residues = list(G.nodes())  # Extract nodes (residue IDs) from the network

# Function to create adjacency matrix
def create_adjacency_matrix(G, residues):
    """Creates an adjacency matrix from the interaction network."""
    residues = list(set(residues))  # Ensure uniqueness
    adjacency_matrix = nx.to_numpy_array(G, nodelist=residues)
    return adjacency_matrix


# Create adjacency matrix using the corrected residue list
adj_matrix = create_adjacency_matrix(G, residues)

print("Adjacency Matrix:")
print(adj_matrix)

# Save adjacency matrix to file
np.savetxt("adjacency_matrix.txt", adj_matrix, fmt="%d")
```

```
# Draw the network
plt.figure(figsize=(8, 6))
nx.draw(G, with_labels=True, node_color='lightblue', edge_color='gray',␣
  ↪node_size=500, font_size=8)
plt.title("Protein-Residue Interaction Network")
plt.show()
```

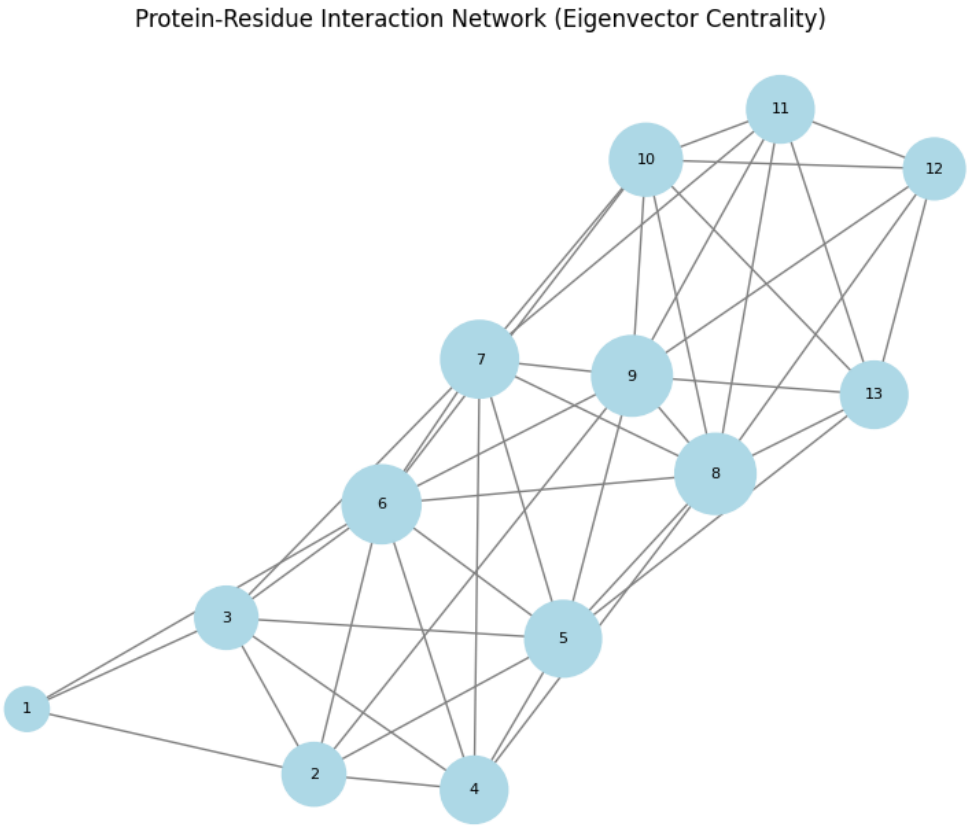
```
Structure exists: '/home/smohtash/a5/pdb6a5j.ent'
Adjacency Matrix:
[[0. 1. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 1. 1. 1. 1. 0. 0. 1. 0. 0. 0. 0.]
 [1. 1. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0.]
 [0. 1. 1. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0.]
 [0. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0. 0.]
 [0. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0.]
 [0. 0. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1.]
 [0. 1. 0. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1.]
 [0. 0. 0. 0. 0. 1. 1. 1. 1. 0. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 0. 1.]
 [0. 0. 0. 0. 1. 0. 0. 1. 1. 1. 1. 1. 0.]]
```



[3]:
```python
def compute_eigenvector_centrality(G):
    """
    Computes the eigenvector centrality of the protein-residue interaction network.

    Parameters:
    G (networkx.Graph): The protein-residue interaction network.

    Returns:
    dict: A dictionary of nodes with their eigenvector centrality scores.
    """
    try:
        centrality = nx.eigenvector_centrality(G, max_iter=1000, tol=1e-6)
        return centrality
    except nx.NetworkXError as e:
        print(f"Eigenvector centrality calculation failed: {e}")
        return None

# Compute eigenvector centrality
eigenvector_centrality = compute_eigenvector_centrality(G)

# Print top residues by eigenvector centrality
print("Top Residues by Eigenvector Centrality:")
sorted_centrality = sorted(eigenvector_centrality.items(), key=lambda item: item[1], reverse=True)
for residue, score in sorted_centrality[:10]:  # Print top 10 residues
    print(f"Residue {residue}: {score:.4f}")

# Visualization: Draw network with node sizes based on eigenvector centrality
plt.figure(figsize=(8, 6))
node_sizes = [5000 * eigenvector_centrality[node] for node in G.nodes()]  # Scale node sizes
nx.draw(G, with_labels=True, node_color='lightblue', edge_color='gray',
        node_size=node_sizes, font_size=8)
plt.title("Protein-Residue Interaction Network (Eigenvector Centrality)")
plt.show()
```

```
Top Residues by Eigenvector Centrality:
Residue 8: 0.3586
Residue 9: 0.3553
Residue 6: 0.3395
Residue 7: 0.3297
Residue 5: 0.3212
Residue 10: 0.2893
Residue 11: 0.2480
```

```
Residue 4: 0.2478
Residue 13: 0.2469
Residue 2: 0.2204
```



[4]:
```
# import numpy as np
# import pennylane as qml
# from pennylane import numpy as pnp

# # Define adjacency matrix A for a sample graph (example)
# A = adj_matrix
# n = A.shape[0]  # Number of nodes
# d = np.sum(A, axis=1)  # Degree sequence
# d_norm = d / np.linalg.norm(d)  # Normalized degree vector
# d_matrix = np.outer(d_norm, d_norm)  # dd^T matrix

# # Define penalty parameters
# P0 = 1 / np.sqrt(n)
# P1 = tau * n

# def construct_qubo(A, P0, P1, tau):
```

```
#     """Constructs QUBO matrix Q."""
#     n = A.shape[0]
#     C = (1 - 2 * tau) * np.eye(n) + np.ones((n, n)) - np.eye(n)
#     Q = -P0 * (A @ d_matrix @ A + A @ d_matrix @ A @ A) + P1 * C
#     return Q

# # Define tau (number of most influential nodes to identify)
# tau = 1
# Q = construct_qubo(A, P0, P1, tau)

# # Convert QUBO to cost Hamiltonian
# coeffs = []
# ops = []
# for i in range(n):
#     coeffs.append(Q[i, i])
#     ops.append(qml.PauliZ(i))
# for i in range(n):
#     for j in range(i + 1, n):
#         coeffs.append(Q[i, j])
#         ops.append(qml.PauliZ(i) @ qml.PauliZ(j))
# cost_hamiltonian = qml.Hamiltonian(coeffs, ops)

# # Define mixer Hamiltonian
# mixer_hamiltonian = qml.Hamiltonian(
#     [1.0] * n, [qml.PauliX(i) for i in range(n)]
# )

# # Define QAOA circuit with sampling enabled
# dev = qml.device("default.qubit", wires=n, shots=1024)

# @qml.qnode(dev)
# def qaoa_circuit(params):
#     """QAOA circuit for optimizing the QUBO problem."""
#     for i in range(n):
#         qml.Hadamard(wires=i)
#     for p in range(len(params) // 2):
#         qml.templates.ApproxTimeEvolution(cost_hamiltonian, params[2 * p], 1)
#         qml.templates.ApproxTimeEvolution(mixer_hamiltonian, params[2 * p +␣
 ↪1], 1)
#     return qml.expval(cost_hamiltonian)

# # Define cost function for optimization
# def cost_function(params):
#     return qaoa_circuit(params)

# # Optimize using classical method
# opt = qml.GradientDescentOptimizer(stepsize=0.1)
```

```
# params = pnp.random.uniform(-np.pi, np.pi, size=(6,), requires_grad=True)  # For p=3
# for _ in range(100):
#     params = opt.step(cost_function, params)

# # Print results
# print("Optimized parameters:", params)
# print("Final cost value:", cost_function(params))
```

[5]:
```
# import numpy as np

# # Run the optimized QAOA circuit
# binary_solution = qaoa_circuit(params)

# # Ensure binary_solution is an array
# binary_solution = np.array(binary_solution)

# # If it's a scalar (0-D array), reshape it into a 1-D array
# if binary_solution.ndim == 0:
#     binary_solution = np.array([binary_solution])

# # Debugging: Print the raw output
# print("Raw output from QAOA circuit:", binary_solution)
# print("Binary solution shape:", binary_solution.shape)

# # Convert -1,1 Pauli-Z output to binary (0,1)
# binary_solution = [(1 if x == -1 else 0) for x in binary_solution.flatten()]

# # Extract influential nodes
# selected_nodes = [i for i, value in enumerate(binary_solution) if value == 1]

# # Print the extracted nodes
# print("Most influential nodes:", selected_nodes)
```

[6]:
```
pip install dwave-ocean-sdk
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: dwave-ocean-sdk in ./.local/lib/python3.11/site-
packages (8.2.0)
Requirement already satisfied: dimod==0.12.18 in ./.local/lib/python3.11/site-
packages (from dwave-ocean-sdk) (0.12.18)
Requirement already satisfied: dwave-cloud-client==0.13.3 in
./.local/lib/python3.11/site-packages (from dwave-ocean-sdk) (0.13.3)
Requirement already satisfied: dwave-gate==0.3.3 in
./.local/lib/python3.11/site-packages (from dwave-ocean-sdk) (0.3.3)
Requirement already satisfied: dwave-hybrid==0.6.13 in
./.local/lib/python3.11/site-packages (from dwave-ocean-sdk) (0.6.13)
```

```
Requirement already satisfied: dwave-inspector==0.5.2 in
./.local/lib/python3.11/site-packages (from dwave-ocean-sdk) (0.5.2)
Requirement already satisfied: dwave-networkx==0.8.16 in
./.local/lib/python3.11/site-packages (from dwave-ocean-sdk) (0.8.16)
Requirement already satisfied: dwave-optimization==0.5.1 in
./.local/lib/python3.11/site-packages (from dwave-ocean-sdk) (0.5.1)
Requirement already satisfied: dwave-preprocessing==0.6.7 in
./.local/lib/python3.11/site-packages (from dwave-ocean-sdk) (0.6.7)
Requirement already satisfied: dwave-samplers==1.4.0 in
./.local/lib/python3.11/site-packages (from dwave-ocean-sdk) (1.4.0)
Requirement already satisfied: dwave-system==1.29.0 in
./.local/lib/python3.11/site-packages (from dwave-ocean-sdk) (1.29.0)
Requirement already satisfied: dwavebinarycsp==0.3.1 in
./.local/lib/python3.11/site-packages (from dwave-ocean-sdk) (0.3.1)
Requirement already satisfied: minorminer==0.2.17 in
./.local/lib/python3.11/site-packages (from dwave-ocean-sdk) (0.2.17)
Requirement already satisfied: penaltymodel==1.2.0 in
./.local/lib/python3.11/site-packages (from dwave-ocean-sdk) (1.2.0)
Requirement already satisfied: numpy>=1.17.3 in ./.local/lib/python3.11/site-
packages (from dimod==0.12.18->dwave-ocean-sdk) (1.26.4)
Requirement already satisfied: requests<3,>=2.25 in /apps/spack/negishi/apps/ana
conda/2024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from
requests[socks]<3,>=2.25->dwave-cloud-client==0.13.3->dwave-ocean-sdk) (2.31.0)
Requirement already satisfied: urllib3<3,>=1.26 in ./.local/lib/python3.11/site-
packages (from dwave-cloud-client==0.13.3->dwave-ocean-sdk) (2.4.0)
Requirement already satisfied: pydantic<3,>=2 in ./.local/lib/python3.11/site-
packages (from dwave-cloud-client==0.13.3->dwave-ocean-sdk) (2.9.2)
Requirement already satisfied: homebase<2,>=1.0 in ./.local/lib/python3.11/site-
packages (from dwave-cloud-client==0.13.3->dwave-ocean-sdk) (1.0.1)
Requirement already satisfied: click<9,>=7.0 in /apps/spack/negishi/apps/anacond
a/2024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from dwave-
cloud-client==0.13.3->dwave-ocean-sdk) (8.1.7)
Requirement already satisfied: python-dateutil<3,>=2.7 in
./.local/lib/python3.11/site-packages (from dwave-cloud-client==0.13.3->dwave-
ocean-sdk) (2.9.0.post0)
Requirement already satisfied: plucky<0.5,>=0.4.3 in
./.local/lib/python3.11/site-packages (from dwave-cloud-client==0.13.3->dwave-
ocean-sdk) (0.4.3)
Requirement already satisfied: diskcache<6,>=5.2.1 in
./.local/lib/python3.11/site-packages (from dwave-cloud-client==0.13.3->dwave-
ocean-sdk) (5.6.3)
Requirement already satisfied: packaging>=19 in /apps/spack/negishi/apps/anacond
a/2024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from dwave-
cloud-client==0.13.3->dwave-ocean-sdk) (23.1)
Requirement already satisfied: werkzeug<4,>=2.2 in /apps/spack/negishi/apps/anac
onda/2024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from dwave-
cloud-client==0.13.3->dwave-ocean-sdk) (2.2.3)
Requirement already satisfied: typing-extensions<5,>=4.5.0 in
```

```
./.local/lib/python3.11/site-packages (from dwave-cloud-client==0.13.3->dwave-
ocean-sdk) (4.13.2)
Requirement already satisfied: authlib<2,>=1.2 in ./.local/lib/python3.11/site-
packages (from dwave-cloud-client==0.13.3->dwave-ocean-sdk) (1.5.1)
Requirement already satisfied: importlib_metadata>=5.0.0 in /apps/spack/negishi/
apps/anaconda/2024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from
dwave-cloud-client==0.13.3->dwave-ocean-sdk) (7.0.1)
Requirement already satisfied: orjson>=3.10 in ./.local/lib/python3.11/site-
packages (from dwave-cloud-client==0.13.3->dwave-ocean-sdk) (3.10.15)
Requirement already satisfied: networkx in ./.local/lib/python3.11/site-packages
(from dwave-hybrid==0.6.13->dwave-ocean-sdk) (3.4.2)
Requirement already satisfied: Flask<4,>=2.2 in /apps/spack/negishi/apps/anacond
a/2024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from dwave-
inspector==0.5.2->dwave-ocean-sdk) (2.2.5)
Requirement already satisfied: scipy>=1.7.3 in ./.local/lib/python3.11/site-
packages (from dwave-system==1.29.0->dwave-ocean-sdk) (1.15.3)
Requirement already satisfied: fasteners>=0.15 in ./.local/lib/python3.11/site-
packages (from minorminer==0.2.17->dwave-ocean-sdk) (0.19)
Requirement already satisfied: cryptography in /apps/spack/negishi/apps/anaconda
/2024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from
authlib<2,>=1.2->dwave-cloud-client==0.13.3->dwave-ocean-sdk) (42.0.2)
Requirement already satisfied: Jinja2>=3.0 in /apps/spack/negishi/apps/anaconda/
2024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from
Flask<4,>=2.2->dwave-inspector==0.5.2->dwave-ocean-sdk) (3.1.3)
Requirement already satisfied: itsdangerous>=2.0 in /apps/spack/negishi/apps/ana
conda/2024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from
Flask<4,>=2.2->dwave-inspector==0.5.2->dwave-ocean-sdk) (2.0.1)
Requirement already satisfied: zipp>=0.5 in /apps/spack/negishi/apps/anaconda/20
24.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from
importlib_metadata>=5.0.0->dwave-cloud-client==0.13.3->dwave-ocean-sdk) (3.17.0)
Requirement already satisfied: annotated-types>=0.6.0 in
./.local/lib/python3.11/site-packages (from pydantic<3,>=2->dwave-cloud-
client==0.13.3->dwave-ocean-sdk) (0.7.0)
Requirement already satisfied: pydantic-core==2.23.4 in
./.local/lib/python3.11/site-packages (from pydantic<3,>=2->dwave-cloud-
client==0.13.3->dwave-ocean-sdk) (2.23.4)
Requirement already satisfied: six>=1.5 in ./.local/lib/python3.11/site-packages
(from python-dateutil<3,>=2.7->dwave-cloud-client==0.13.3->dwave-ocean-sdk)
(1.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /apps/spack/negishi/a
pps/anaconda/2024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from
requests<3,>=2.25->requests[socks]<3,>=2.25->dwave-cloud-client==0.13.3->dwave-
ocean-sdk) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /apps/spack/negishi/apps/anaconda
/2024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from
requests<3,>=2.25->requests[socks]<3,>=2.25->dwave-cloud-client==0.13.3->dwave-
ocean-sdk) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in /apps/spack/negishi/apps/an
```

```
aconda/2024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from
requests<3,>=2.25->requests[socks]<3,>=2.25->dwave-cloud-client==0.13.3->dwave-
ocean-sdk) (2024.2.2)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /apps/spack/negishi/app
s/anaconda/2024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from
requests[socks]<3,>=2.25->dwave-cloud-client==0.13.3->dwave-ocean-sdk) (1.7.1)
Requirement already satisfied: MarkupSafe>=2.1.1 in /apps/spack/negishi/apps/ana
conda/2024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from
werkzeug<4,>=2.2->dwave-cloud-client==0.13.3->dwave-ocean-sdk) (2.1.3)
Requirement already satisfied: cffi>=1.12 in /apps/spack/negishi/apps/anaconda/2
024.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from
cryptography->authlib<2,>=1.2->dwave-cloud-client==0.13.3->dwave-ocean-sdk)
(1.16.0)
Requirement already satisfied: pycparser in /apps/spack/negishi/apps/anaconda/20
24.02-py311-gcc-8.5.0-lr57z2f/lib/python3.11/site-packages (from
cffi>=1.12->cryptography->authlib<2,>=1.2->dwave-cloud-client==0.13.3->dwave-
ocean-sdk) (2.21)
DEPRECATION: qiskit-nature 0.5.0 has a non-standard dependency specifier
qiskit-terra>=0.22.*. pip 24.0 will enforce this behaviour change. A possible
replacement is to upgrade to a newer version of qiskit-nature or contact the
author to suggest that they release a version with a conforming dependency
specifiers. Discussion can be found at
https://github.com/pypa/pip/issues/12063
Note: you may need to restart the kernel to use updated packages.
```

[7]:
```python
import networkx as nx
import numpy as np
from dimod import BinaryQuadraticModel, SimulatedAnnealingSampler

# Define constants

A = adj_matrix
n = len(A)
tau = 5  # Desired number of top nodes
P0 = 1 / np.sqrt(n)  # Based on the paper
P1 = 10 * n          # Stronger penalty to enforce the constraint

def calculate_d_hat(adj_matrix):
    d = np.sum(adj_matrix, axis=1)
    d_hat = d / np.linalg.norm(d)
    return d_hat

d_hat = calculate_d_hat(A)
```

```
# Construct C matrix
I = np.eye(n)
U = np.ones((n, n)) - I
C = (1 - 2 * tau) * I + U

# Construct Q matrix
Q = -P0 * (A @ np.outer(d_hat, d_hat) @ A) \
    - P0 * (A @ np.outer(d_hat, d_hat) @ A @ A) \
    + P1 * C

# Convert to BQM
bqm = BinaryQuadraticModel.from_numpy_matrix(Q)

# Simulated annealing
sampler = SimulatedAnnealingSampler()
response = sampler.sample(bqm, num_reads=10000, beta_range=(0.1, 4.0))

# Filter solutions with exactly tau selected nodes
valid_solutions = []
for sample, energy in response.data(['sample', 'energy']):
    if sum(sample.values()) == tau:
        valid_solutions.append((sample, energy))

if valid_solutions:
    best_sample, best_energy = min(valid_solutions, key=lambda x: x[1])
    top_nodes = [node + 1 for node, val in best_sample.items() if val == 1]

    print("Best valid sample:", best_sample)
    print("Energy:", best_energy)
    print("Top nodes:", top_nodes)
else:
    print("No valid solution with exactly   selected nodes.")
```

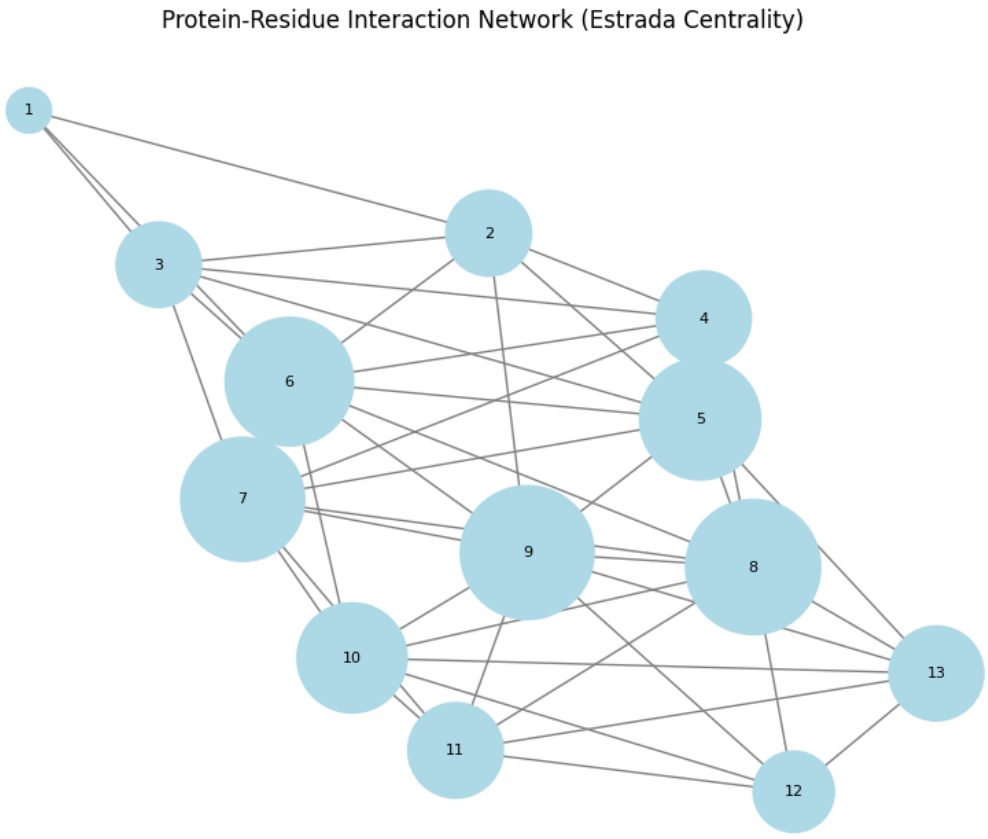
```
/tmp/ipykernel_2797332/628357712.py:31: DeprecationWarning:
BQM.from_numpy_matrix(M) is deprecated since dimod 0.10.0 and will be removed in
0.12.0. Use BQM(M, "BINARY") instead.
  bqm = BinaryQuadraticModel.from_numpy_matrix(Q)
```

```
Best valid sample: {0: 0, 1: 0, 2: 0, 3: 0, 4: 1, 5: 1, 6: 1, 7: 1, 8: 1, 9: 0,
10: 0, 11: 0, 12: 0}
Energy: -3593.210247357017
Top nodes: [5, 6, 7, 8, 9]
```

[8]:
```
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
from scipy.linalg import expm
```

```python
def compute_estrada_centrality_manual(G):
    """
    Computes Estrada centrality by evaluating the diagonal of the matrix␣
 ↪exponential of A.

    Parameters:
    G (networkx.Graph): The protein-residue interaction network.

    Returns:
    dict: A dictionary of nodes with their Estrada centrality scores.
    """
    A = nx.to_numpy_array(G)  # Adjacency matrix
    E = expm(A)               # Matrix exponential e^A
    estrada_scores = np.diag(E)  # Extract diagonal elements

    return {node: estrada_scores[i] for i, node in enumerate(G.nodes())}

# --- Compute Estrada Centrality ---
estrada_centrality = compute_estrada_centrality_manual(G)

# --- Print Top Residues by Estrada Centrality ---
print("Top Residues by Estrada Centrality:")
sorted_estrada = sorted(estrada_centrality.items(), key=lambda item: item[1],␣
 ↪reverse=True)
for residue, score in sorted_estrada[:10]:  # Top 10
    print(f"Residue {residue}: {score:.4f}")

# --- Visualization: Network Colored by Estrada Centrality ---
plt.figure(figsize=(8, 6))
node_sizes = [5000 * estrada_centrality[node] / max(estrada_centrality.
 ↪values()) for node in G.nodes()]
nx.draw(G, with_labels=True, node_color='lightblue', edge_color='gray',
        node_size=node_sizes, font_size=8)
plt.title("Protein-Residue Interaction Network (Estrada Centrality)")
plt.show()
```

```
Top Residues by Estrada Centrality:
Residue 8: 175.3744
Residue 9: 172.0556
Residue 6: 158.6443
Residue 7: 147.8283
Residue 5: 141.4522
Residue 10: 116.5128
Residue 11: 87.7037
Residue 4: 86.8848
Residue 13: 86.4379
Residue 2: 70.7401
```



[9]:
```python
import networkx as nx
import numpy as np
from dimod import BinaryQuadraticModel, SimulatedAnnealingSampler

# --- Input: Adjacency matrix ---
A = adj_matrix  # Use your preloaded or computed adjacency matrix here
n = len(A)
tau = 1
P0 = 1 / np.sqrt(n)
P1 = 100 * n

# --- Step 1: Degree normalization ---
def calculate_d_hat(adj_matrix):
    d = np.sum(adj_matrix, axis=1)
    d_hat = d / np.linalg.norm(d)
    return d_hat

d_hat = calculate_d_hat(A)
```

```
# --- Step 2: Construct matrix exponential approximation E = I + A + 0.5 A^2 +
 ↪(1/6) A^3 ---
A2 = A @ A
A3 = A2 @ A
E = np.eye(n) + A + 0.5 * A2 + (1/6) * A3

# --- Step 3: Build low-rank Estrada projection: E * d_hat * d_hat^T * E ---
d_outer = np.outer(d_hat, d_hat)
projection = E @ d_outer @ E  # Rank-1 projection of spectral walk weights

# --- Step 4: Constraint matrix ---
I = np.eye(n)
U = np.ones((n, n)) - I
C = (1 - 2 * tau) * I + U

# --- Step 5: Final QUBO matrix ---
Q = -P0 * projection + P1 * C

# --- Step 6: Convert QUBO to BQM ---
bqm = BinaryQuadraticModel.from_numpy_matrix(Q)

# --- Step 7: Simulated annealing to solve QUBO ---
sampler = SimulatedAnnealingSampler()
response = sampler.sample(bqm, num_reads=10000, beta_range=(0.1, 4.0))

# --- Step 8: Filter valid  -sparse solutions ---
valid_solutions = []
for sample, energy in response.data(['sample', 'energy']):
    if sum(sample.values()) == tau:
        valid_solutions.append((sample, energy))

if valid_solutions:
    best_sample, best_energy = min(valid_solutions, key=lambda x: x[1])
    top_nodes = [node + 1 for node, val in best_sample.items() if val == 1]
    print("Best valid sample:", best_sample)
    print("Energy:", best_energy)
    print("Top nodes (Estrada centrality):", top_nodes)
else:
    print("No valid solution with exactly  selected nodes.")
```

```
/tmp/ipykernel_2797332/3288079529.py:38: DeprecationWarning:
BQM.from_numpy_matrix(M) is deprecated since dimod 0.10.0 and will be removed in
0.12.0. Use BQM(M, "BINARY") instead.
  bqm = BinaryQuadraticModel.from_numpy_matrix(Q)
```

```
Best valid sample: {0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 1, 8: 0, 9: 0,
10: 0, 11: 0, 12: 0}
Energy: -1630.3532767205543
```

```
Top nodes (Estrada centrality): [8]
```

[ ]: