Quantum Information Science HW P2

Harris A. Ransom

11/11/24

In [2]:
```
# Imports
import numpy as np
```

Question 1

Each Tanner graph is considered a **regular bipartite** graph, since it can be divided into two sets of nodes that do not have any intraconnecting edges within their respective sets. For a random (3;6) biregular Tanner graph, we can compute the **degree distribution generating polynomials** as:

- $\lambda(x) = \sum_{d=1}^{3} \Lambda_d x^{d-1} = x^2$
- $\rho(x) = \sum_{d=1}^{6} P_d x^{d-1} = x^5$

Question 2

Question 2a

Successful decoding condition: $\epsilon\lambda\left(1-\rho(1-x)\right) \leq x \text{ for } 0 < x < \epsilon$

We can express $(1-\rho(1-x))$ using the above polynomial as $\alpha = (1-(1-x)^5)$

From this, we get $\lambda(\alpha) = \left(1-(1-x)^5\right)^2$. Note that, from close observation, $\lambda(\alpha)$ is a CDF function, potentially of the degree distribution.

Plugging this back into the decoding condition, we get: $\epsilon\left(1-(1-x)^5\right)^2 \leq x \text{ for } 0 < x < \epsilon$

$\frac{1}{x} \cdot \left(1-(1-x)^5\right)^2 \leq \frac{1}{\epsilon} \text{ for } 0 < x < \epsilon$

$x^9 - 10x^8 + 45x^7 - 120x^6 + 210x^5 - 250x^4 + 200x^3 - 100x^2 + 25x \leq \frac{1}{\epsilon} \text{ for } 0 < x < \epsilon$

Since ϵ represents the maximum channel erasure rate, we want it to fall between 0 and 1. Therefore, we can adjust our condition as:
$x^9 - 10x^8 + 45x^7 - 120x^6 + 210x^5 - 250x^4 + 200x^3 - 100x^2 + 25x \leq \frac{1}{\epsilon} \text{ for } 0 < x < \epsilon \leq 1$

We can rewrite the condition into a better form:
$\epsilon \cdot (x^9 - 10x^8 + 45x^7 - 120x^6 + 210x^5 - 250x^4 + 200x^3 - 100x^2 + 25x) \leq 1 \text{ for } 0 < x < \epsilon \leq 1$

From this condition, for values of ϵ between 0 and 1, we find that the theoretical maximum erasure rate ϵ that satisfies the condition is $\epsilon \approx 0.4295$

Question 2b

In [3]:
```python
# Gets the degree of a check node
def getCheckNodeDegree(A, node):
        degree = np.sum(A[:, node])
        return degree

# Set up adjacency matrix and message
print("Setting up Tanner graph parameters...")
VARIABLE_NODES = 2000
CHECK_NODES = 1000
A = np.zeros((VARIABLE_NODES, CHECK_NODES)) # Adjacency matrix
m = np.zeros(VARIABLE_NODES) # Message (all zeros)

# Distribute connections between variable and check nodes
# Rows: 3 (degree 3)
# Cols: 6 (check 6)
# TODO: Figure out why generation hangs on node 1999
```

```
print("Generating (3;6) Tanner graph...")
colCounts = np.zeros(CHECK_NODES) # Tracks how many columns are full
for i in range(0, VARIABLE_NODES):
        chosenCols = [] # Used to check for duplicates
        for k in range(3):
                # Get random check node edge (column) to add
                randCol = np.random.choice(np.arange(CHECK_NODES))
                while ((colCounts[randCol] == 6) or (randCol in chosenCols)):
                        randCol = np.random.choice(np.arange(CHECK_NODES))

                #print(f"Finished picking column {k} for node {i}")
                # Add chosen column to duplicate tracker, column count tracker, and adjacency matrix
                chosenCols.append(randCol)
                colCounts[randCol] += 1 # TODO: Fix colCounts
                if (A[i, randCol] == 1):
                        raise ValueError(f"Duplicate for variable node {i}")
                A[i, randCol] = 1

# Verify that distribution criteria is met for Tanner Graph
print("Checking graph criteria...")
for i in range(0, VARIABLE_NODES):
        rowSum = sum(A[i, :])
        if (rowSum != 3):
                raise ValueError(f"Incorrect degree count for variable node {i} ({rowSum})")
for i in range(0, CHECK_NODES):
        nodeDegree = getCheckNodeDegree(A, i)
        if (nodeDegree != 6):
                raise ValueError(f"Incorrect degree count for check node {i} ({nodeDegree})")
print("All criteria passed")

# Create base copy of A to restore after each sim run
A_start = A

# Simulates different erasure rates
# Erasures simulated by -1 value

# for run in range(10):
# epsilons = []

for epsilon in np.arange(0.005, 1, 0.005):
        print(f"Simulating erasure rate epsilon={epsilon}")
```

```
# Reset adjacency matrix
A = A_start

# Erase nodes at rate epsilon
m = np.random.choice([-1, 0], size=(VARIABLE_NODES), p=[epsilon, 1-epsilon])

# Iterate until there are no more nodes to clean up
continueFlag = True
while (continueFlag):
        continueFlag = False

        # Erase received variable nodes
        k = 0
        while (k < len(m)):
                if (m[k] == 0):
                        A = np.delete(A, k, axis=0) # Remove variable node row from adjacency matrix
                        m = np.delete(m, k, axis=0) # Remove received node from vector
                else:
                        k += 1

        # Check if m is empty
        if (m.size == 0):
                continueFlag = False
                break

        # Clean up degree-one check nodes
        for j in range(0, CHECK_NODES):
                if (getCheckNodeDegree(A, j) == 1):
                        # Get index of unique neighbor
                        ind = np.argwhere(A[:, j] == 1)

                        # Substitute 0 into unique neighbor variable node
                        m[ind] = 0

                        # Set flag accordingly
                        continueFlag = True

# Determine whether or not erasure recovery fails based on continue flag
# If m is empty, all nodes were recovered
# If m is not empty, recovery was not successful
```

```
        if (m.size > 0):
                print(f"Max erasure rate simulated: {epsilon}")
                break
```

```
Setting up Tanner graph parameters...
Generating (3;6) Tanner graph...
Checking graph criteria...
All criteria passed
Simulating erasure rate epsilon=0.005
Simulating erasure rate epsilon=0.01
Simulating erasure rate epsilon=0.015
Simulating erasure rate epsilon=0.02
Simulating erasure rate epsilon=0.025
Simulating erasure rate epsilon=0.030000000000000002
Simulating erasure rate epsilon=0.034999999999999996
Simulating erasure rate epsilon=0.04
Simulating erasure rate epsilon=0.045
Simulating erasure rate epsilon=0.049999999999999996
Simulating erasure rate epsilon=0.055
Simulating erasure rate epsilon=0.06
Simulating erasure rate epsilon=0.065
Simulating erasure rate epsilon=0.07
Simulating erasure rate epsilon=0.07500000000000001
Simulating erasure rate epsilon=0.08
Simulating erasure rate epsilon=0.085
Simulating erasure rate epsilon=0.09000000000000001
Simulating erasure rate epsilon=0.095
Simulating erasure rate epsilon=0.1
Simulating erasure rate epsilon=0.10500000000000001
Simulating erasure rate epsilon=0.11
Simulating erasure rate epsilon=0.115
Simulating erasure rate epsilon=0.12000000000000001
Simulating erasure rate epsilon=0.125
Simulating erasure rate epsilon=0.13
Simulating erasure rate epsilon=0.135
Simulating erasure rate epsilon=0.14
Simulating erasure rate epsilon=0.14500000000000002
Simulating erasure rate epsilon=0.15
Simulating erasure rate epsilon=0.155
Simulating erasure rate epsilon=0.16
Simulating erasure rate epsilon=0.165
Simulating erasure rate epsilon=0.17
Simulating erasure rate epsilon=0.17500000000000002
Simulating erasure rate epsilon=0.18000000000000002
Simulating erasure rate epsilon=0.185
```

```
Simulating erasure rate epsilon=0.19
Simulating erasure rate epsilon=0.195
Simulating erasure rate epsilon=0.2
Simulating erasure rate epsilon=0.20500000000000002
Simulating erasure rate epsilon=0.21000000000000002
Simulating erasure rate epsilon=0.215
Simulating erasure rate epsilon=0.22
Simulating erasure rate epsilon=0.225
Simulating erasure rate epsilon=0.23
Simulating erasure rate epsilon=0.23500000000000001
Simulating erasure rate epsilon=0.24000000000000002
Simulating erasure rate epsilon=0.245
Simulating erasure rate epsilon=0.25
Simulating erasure rate epsilon=0.255
Simulating erasure rate epsilon=0.26
Simulating erasure rate epsilon=0.265
Simulating erasure rate epsilon=0.27
Simulating erasure rate epsilon=0.275
Simulating erasure rate epsilon=0.28
Simulating erasure rate epsilon=0.28500000000000003
Simulating erasure rate epsilon=0.29000000000000004
Simulating erasure rate epsilon=0.295
Simulating erasure rate epsilon=0.3
Simulating erasure rate epsilon=0.305
Simulating erasure rate epsilon=0.31
Simulating erasure rate epsilon=0.315
Simulating erasure rate epsilon=0.32
Simulating erasure rate epsilon=0.325
Simulating erasure rate epsilon=0.33
Simulating erasure rate epsilon=0.335
Simulating erasure rate epsilon=0.34
Simulating erasure rate epsilon=0.34500000000000003
Simulating erasure rate epsilon=0.35000000000000003
Simulating erasure rate epsilon=0.35500000000000004
Simulating erasure rate epsilon=0.36
Simulating erasure rate epsilon=0.365
Simulating erasure rate epsilon=0.37
Simulating erasure rate epsilon=0.375
Simulating erasure rate epsilon=0.38
Simulating erasure rate epsilon=0.385
Simulating erasure rate epsilon=0.39
```

```
Simulating erasure rate epsilon=0.395
Simulating erasure rate epsilon=0.4
Simulating erasure rate epsilon=0.405
Simulating erasure rate epsilon=0.41000000000000003
Simulating erasure rate epsilon=0.41500000000000004
Simulating erasure rate epsilon=0.42000000000000004
Simulating erasure rate epsilon=0.425
Simulating erasure rate epsilon=0.43
Simulating erasure rate epsilon=0.435
Max erasure rate simulated: 0.435
```