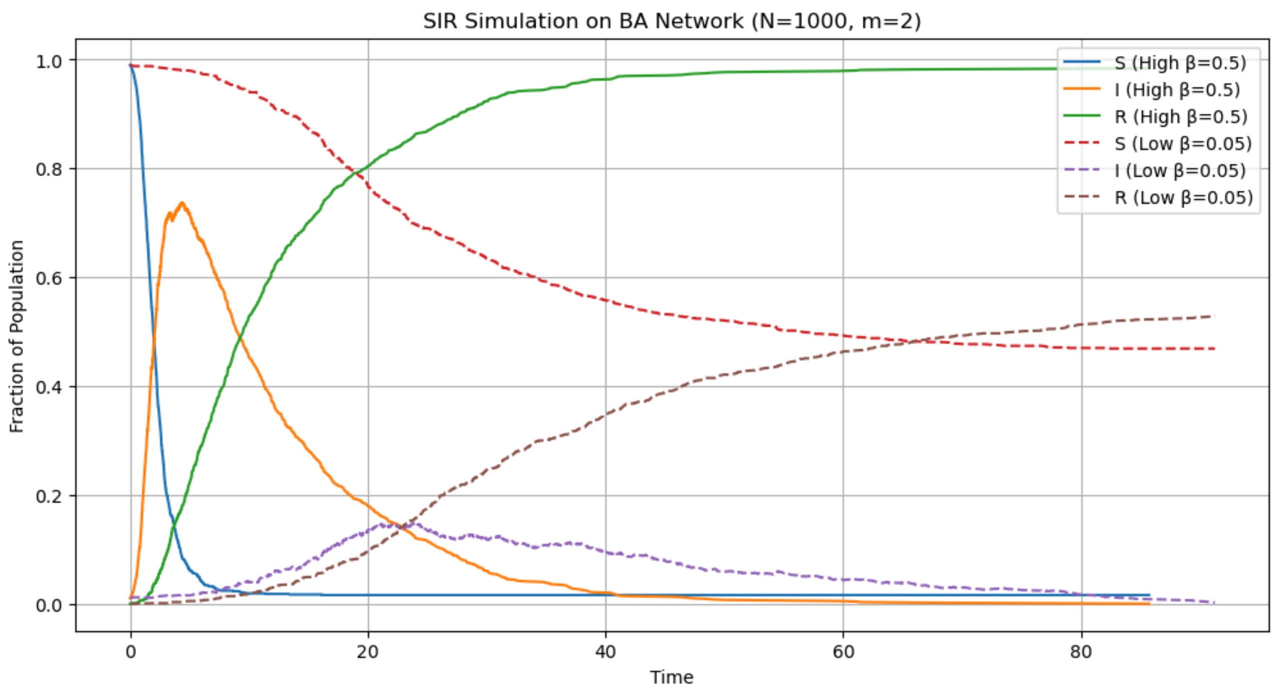
Sunil Thapa
W0779269

1.



2. Part 2:
   - $\langle k \rangle$= 3.992, $\langle k^2 \rangle$= 40.25, λ_c = 0.1101
   - The threshold for Transmission rate 0.5 is: 5.0 >> 0.1101 ($\lambda_{high} >> \lambda_c$). The effective spreading rate is nearly 45 times the theoretical threshold. This overwhelming margin confirms the expectation that the high-spreading rate would result in a major, global outbreak. The simulation should show the infection fraction (*I(t)*) spiking rapidly and nearly all susceptible nodes moving to the recovered state (*R(t)*), consistent with the theoretical prediction for a highly contagious process in a susceptible network.
   - The threshold for Transmission rate 0.05 is: 0.5 > 0.1101 ($\lambda_{low} > \lambda_c$). The simulation results show a localized spread where the infection quickly dies out. This discrepancy arises because, in finite scale-free networks, the theoretical threshold predicts possible sustained spread, but low spreading rates are highly susceptible to stochastic extinction; the infection spreads too slowly to reliably navigate the path through the crucial, high-degree hub nodes before the initial infected nodes recover. Thus, while the network is structurally vulnerable (low λ_c), the low dynamic rate of λ_{low}=0.5 fails to overcome the inherent randomness of the simulation, resulting in a localized cluster instead of a large outbreak.

3. Part 3:
   - Original size of Giant Component: 1000, size of largest component after removing the top 10% of hubs: 616
   - The removal of just 10% of the total nodes resulted in a 38.4% reduction in the size of the Giant Component. The dramatic drop in the size of the Giant Component confirms the theoretical prediction regarding the fragility of scale-free networks against targeted attacks. Barabási-Albert networks are defined by their power-law degree distribution, meaning they possess a small

number of extremely highly connected nodes (hubs) and many low-degree nodes. These hubs are the structural glue of the network, creating the short path lengths that define the large connected component.

i. Impact of Targeted Removal: By intentionally removing the 100 highest-degree nodes, we targeted the precise locations that hold the network together. The removal of these few critical bridges causes the rest of the network to rapidly fragment into smaller, isolated clusters that are no longer part of the largest component.
ii. Comparison to Random Failure: In contrast, removing 10% of nodes randomly would likely remove only low-degree nodes, leaving the crucial hubs intact and resulting in a much smaller decrease in GC size. The observed result—where a 10% node removal shatters nearly 40% of the connections to the GC – is the defining characteristic of a scale-free network's vulnerability: robust against random failures, yet extremely vulnerable to focused attacks.

homework-3

November 21, 2025

[9]:
```
import networkx as nx
import EoN
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
```

[14]:
```
# Simulating SIR Model
N = 1000
m = 2
G = nx.barabasi_albert_graph(N, m)
```

[15]:
```
mu = 0.1
initial_fraction_infected = 0.01
initial_infecteds = np.random.choice(G.nodes(),␣
 ↪size=int(N*initial_fraction_infected), replace=False)

# Run the simulation
def run_sir(beta_value):
    t, S, I, R = EoN.fast_SIR(G, tau=beta_value, gamma=mu,␣
 ↪initial_infecteds=initial_infecteds,  tmax=100)
    S_fractions = np.array(S) / N
    I_fractions = np.array(I) / N
    R_fractions = np.array(R) / N
    return t, S_fractions, I_fractions, R_fractions

# High spreading rate
beta_high = 0.5
t_high, S_high, I_high, R_high = run_sir(beta_high)

# Low spreading rate
beta_low = 0.05
t_low, S_low, I_low, R_low = run_sir(beta_low)
```

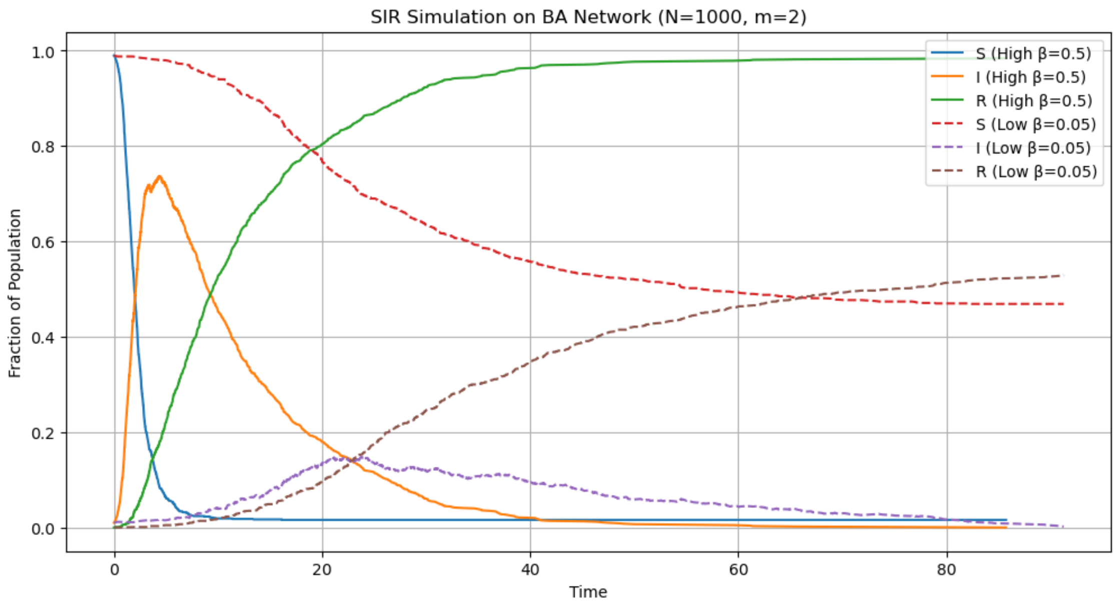
[16]:
```
plt.figure(figsize=(12, 6))

# High spreading lines
plt.plot(t_high, S_high, label="S (High  =0.5)")
```

```python
plt.plot(t_high, I_high, label="I (High  =0.5)")
plt.plot(t_high, R_high, label="R (High  =0.5)")

# Low spreading lines (dashed)
plt.plot(t_low, S_low, '--', label="S (Low  =0.05)")
plt.plot(t_low, I_low, '--', label="I (Low  =0.05)")
plt.plot(t_low, R_low, '--', label="R (Low  =0.05)")

plt.xlabel("Time")
plt.ylabel("Fraction of Population")
plt.title("SIR Simulation on BA Network (N=1000, m=2)")
plt.legend()
plt.grid(True)
plt.show()
```



[17]:
```python
# Calculate the epidemic threshold
degrees = np.array([deg for node, deg in G.degree()])
k_avg = degrees.mean()
k_sq_avg = np.mean(degrees**2)
lambda_c = k_avg / (k_sq_avg - k_avg)
print(f"Epidemic threshold ( c): {lambda_c:.4f}")
```

```
Epidemic threshold ( c): 0.1101
```

[20]:
```python
print("Average degree <k>:", k_avg)
print("Second moment <k^2>:", k_sq_avg)
```

```
Average degree <k>: 3.992
Second moment <k^2>: 40.25
```

[18]:
```
# Calculate effective spreading rates
lambda_high = beta_high / mu
lambda_low = beta_low / mu

print(f"Effective spreading rate ( _high): {lambda_high:.4f}")
print(f"Effective spreading rate ( _low): {lambda_low:.4f}")
```

```
Effective spreading rate ( _high): 5.0000
Effective spreading rate ( _low): 0.5000
```

[19]:
```
# Percolation and Network Robustness
N=1000
m=2
G=nx.barabasi_albert_graph(N,m)

# Identify the top 10% of nodes by Degree Centrality
num_removed = int(N * 0.1)
sorted_nodes = sorted(G.degree(), key=lambda x: x[1], reverse=True)
hubs_to_remove = [node for node, _ in sorted_nodes[:num_removed]]

components = nx.connected_components(G)
original_gc_size = len(max(components, key=len)) if components else 0

# Perform Targeted Attack
G_copy = G.copy()
G_copy.remove_nodes_from(hubs_to_remove)

new_gc_size = len(max(nx.connected_components(G_copy), key=len)) if nx.
↪connected_components(G_copy) else 0

print(f"Original Giant Component Size (GC_orig): {original_gc_size}")
print(f"New Giant Component Size (GC_new): {new_gc_size}")
print("Percentage drop:", 100 * (original_gc_size - new_gc_size) /␣
↪original_gc_size, "%")
```

```
Original Giant Component Size (GC_orig): 1000
New Giant Component Size (GC_new): 616
Percentage drop: 38.4 %
```

[ ]: