Relatório A2

Alice Oliveira e Bruno Andrades

Junho 2025

1 Introdução

Item 2.1 — Modelo de Geração: Erdős–Rényi

Metodologia

- **Número de nós:** 25
- **Probabilidade de conexão:** $p = 0{,}6$
- **Implementação:** manual em Python, utilizando sorteio para determinar quais arestas são criadas.

Visualização da Rede

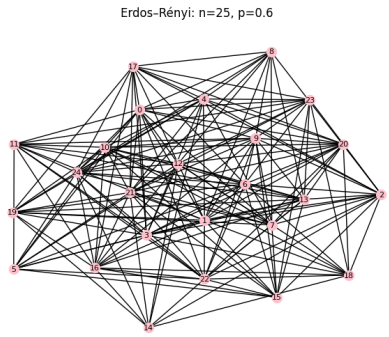
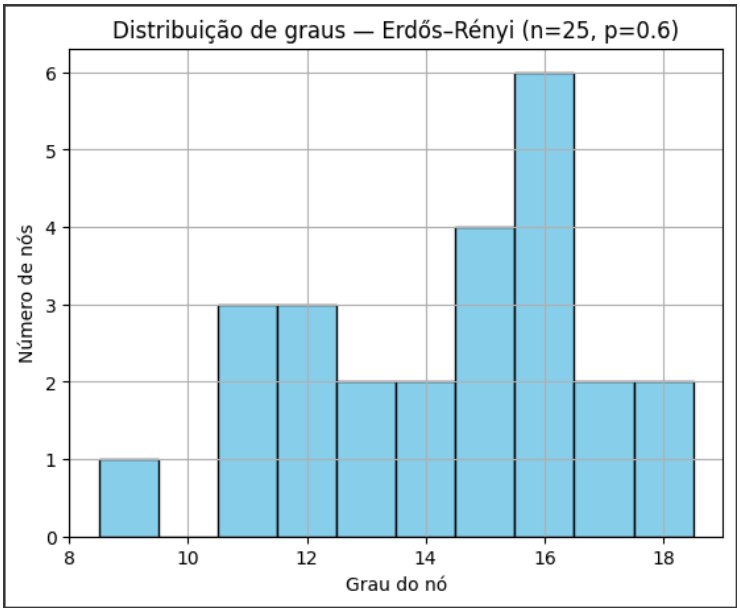


Figure 1: Visualização gerada com o modelo de Erdos-Rényi.

Distribuição de Graus



A distribuição de graus está concentrada em torno do valor médio, como esperado em uma rede aleatória densa.

Métricas da Rede

- **Número de componentes conexas:** 1
- **Diâmetro:** 2
- **Grau médio:** 14.32
- **Coeficiente de clustering médio:** 0.60
- **Densidade:** 0.60

A rede gerada com o modelo de Erdős–Rényi representa uma estrutura densa e homogênea. O grau médio elevado e a alta densidade indicam que a maioria dos nós está conectada entre si. O pequeno diâmetro e o clustering médio mostram que existen muitos caminhos curtos e conexões locais razoáveis, mesmo sendo uma rede aleatória. As métricas nos apresentam uma estrutura bem conectada, mas sem a presença de hubs ou organização hierárquica.

1.1 Código Python para Modelo de Erdos-Rényi

```
import random
import matplotlib.pyplot as plt
import networkx as nx

def erdos_renyi(n, p):

  # grafo vazio
  G = nx.Graph()
  G.add_nodes_from(range(n))  # nós de 0 até n-1

  # percorre todos os pares (i, j) com i < j
  for i in range(n):
      for j in range(i+1, n):
          if random.random() < p:
              G.add_edge(i, j)

  # desenha o grafo
  nx.draw(
      G,
      with_labels=True,
      node_color='pink',
      edge_color='black',
      node_size=100,
      font_size=8
  )
  plt.title(f"Erdos{Rényi: n={n}, p={p}")
  plt.show()
  graus = [grau for _, grau in G.degree()]
  plt.hist(graus, bins=range(min(graus), max(graus)+2), align='left', color='skyblue', edgec
  plt.title(f"Distribuição de graus | Erdos{Rényi (n={n}, p={p})")
  plt.xlabel("Grau do nó")
  plt.ylabel("Número de nós")
  plt.grid(True)
  plt.show()

  print("Número de componentes conexas:", nx.number_connected_components(G))
  print("Diâmetro (se conexo):", nx.diameter(G) if nx.is_connected(G) else "grafo desconexo'
  print(f"Grau médio: {sum(graus)/len(graus):.2f}")
  print(f"Coeficiente de clustering médio: {nx.average_clustering(G):.2f}")
  print(f"Densidade: {nx.density(G):.2f}")


erdos_renyi(25, 0.6)
```

Conclusão

A rede gerada com $p = 0{,}6$ apresenta uma estrutura conectada, mas já com um pouco menos de redundância do que seria esperado com um p mais alto. O diâmetro pequeno e o clustering médio elevados nos mostram que a rede permanece bem conectada, com muitas rotas curtas entre os nós. Apesar da estrutura densa, não existem formação de hubs nem cauda longa na distribuição de graus, como é característico do modelo de Erdős–Rényi. Os graus dos nós se distribuem de forma homogênea, o que mostra o processo aleatório uniforme de geração de arestas. Essa rede serve como base para comparação com modelos mais estruturados ou com dinâmicas de crescimento preferencial.

Item 2.2 — Modelo de Geração: Watts–Strogatz

Metodologia

- **Número de nós:** 25
- **Número de vizinhos k:** 6 (cada nó se conecta aos 3 vizinhos de cada lado)
- **Probabilidade de redirecionamento:** $p = 0{,}6$
- **Implementação:** manual, com construção de anel regular e redirecionamento por sorteio.

Visualização da Rede

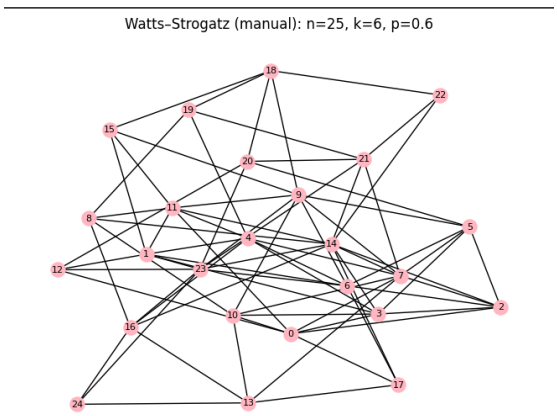


Figure 2: Rede gerada com o modelo de Watts–Strogatz ($n = 25, k = 6, p = 0.6$).

Métricas da Rede

- **Número de componentes conexas:** 1
- **Diâmetro:** 3
- **Caminho médio:** 1.91
- **Grau médio:** 6.00
- **Desvio padrão dos graus:** 1.81
- **Coeficiente de clustering médio:** 0.25
- **Densidade:** 0.25

A rede de Watts–Strogatz combina regularidade com aleatoriedade. O grau médio é constante, e o clustering moderado indica alguma tendência à formação de triângulos. O pequeno caminho médio e diâmetro caracterizam a propriedade de mundo pequeno. O desvio padrão dos graus é baixo, o que reforça a homogeneidade do grau entre os nós. As métricas mostram uma rede conectada, com boa navegabilidade, mas sem concentração de conexões em poucos nós.

Distribuição de Graus

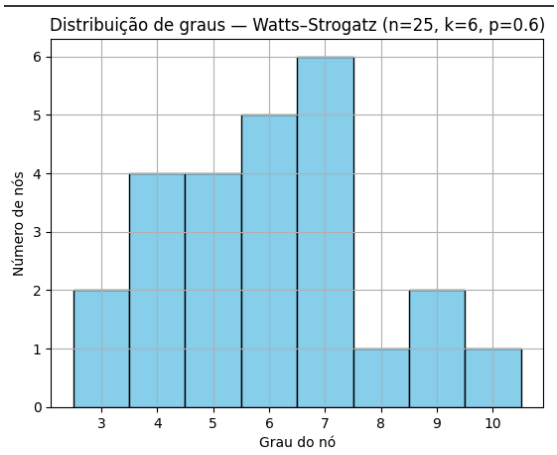


Figure 3: Distribuição de graus da rede gerada com o modelo de Watts–Strogatz.

A distribuição de graus apresenta pouca variação em torno do grau inicial 6, mostrando a aleatoriedade que é introduzida pelo redirecionamento.

1.2 Código Python para Modelo de Wattz-Strogatz

```
import random
import matplotlib.pyplot as plt
import networkx as nx

def watts_strogatz(n, k, p):
    if k % 2 != 0:
        raise ValueError("k deve ser par.")

    G = nx.Graph()
    G.add_nodes_from(range(n))

    # conexões do anel regular
    for i in range(n):
        for j in range(1, k//2 + 1):
            neighbor = (i + j) % n
            G.add_edge(i, neighbor)

    # redirecionamento com probabilidade p
    for (i, j) in list(G.edges()):
        if random.random() < p:
            G.remove_edge(i, j)

            possible_nodes = [n for n in range(n) if n != i and not G.has_edge(i, n)]
            if possible_nodes:
                new_target = random.choice(possible_nodes)
                G.add_edge(i, new_target)
            else:
                G.add_edge(i, j)

    # visualização do grafo
    nx.draw(
        G,
        with_labels=True,
        node_size=150,
        font_size=8,
        edge_color='black',
        node_color='lightpink'
    )
    plt.title(f"Watts{Strogatz (manual): n={n}, k={k}, p={p}")
    plt.show()

    # métricas da rede
    graus = [grau for _, grau in G.degree()]
    print(f"Número de componentes conexas: {nx.number_connected_components(G)}")
```

```
    if nx.is_connected(G):
        print(f"Diâmetro: {nx.diameter(G)}")
        print(f"Caminho médio: {nx.average_shortest_path_length(G):.2f}")
    else:
        print("Atenção: Grafo desconexo { não é possível calcular caminho médio e diâmetro.'

    print(f"Grau médio: {sum(graus)/len(graus):.2f}")
    print(f"Desvio padrão dos graus: {round((sum((x - sum(graus)/len(graus))**2 for x in gra
    print(f"Coeficiente de clustering médio: {nx.average_clustering(G):.2f}")
    print(f"Densidade: {nx.density(G):.2f}")

    # histograma da distribuição de graus
    plt.hist(graus, bins=range(min(graus), max(graus)+2), align='left', color='skyblue', edg
    plt.title(f"Distribuição de graus | Watts{Strogatz (n={n}, k={k}, p={p})")
    plt.xlabel("Grau do nó")
    plt.ylabel("Número de nós")
    plt.grid(True)
    plt.show()

    return G

G_ws = watts_strogatz(n=25, k=6, p=0.6)
```

Conclusão

A rede gerada mostra sinais da transição para o regime de mundo pequeno: o caminho médio é bastante reduzido e o clustering ainda se mantém moderado. O diâmetro pequeno e a estrutura conectada são esperados para esse valor de p. O modelo permite estudar como a inserção de aleatoriedade afeta as propriedades locais e globais da rede.

Item 2.3 — Modelo de Geração: Rede com Comunidades

Metodologia

- **Número de nós:** 100
- **Número de comunidades:** 4
- **Probabilidade de conexão dentro da comunidade:** $p_{in} = 0{,}4$
- **Probabilidade de conexão entre comunidades:** $p_{out} = 0{,}02$
- **Implementação:** manual, utilizando sorteios para definir as arestas dentro e fora das comunidades.

Visualização da Rede

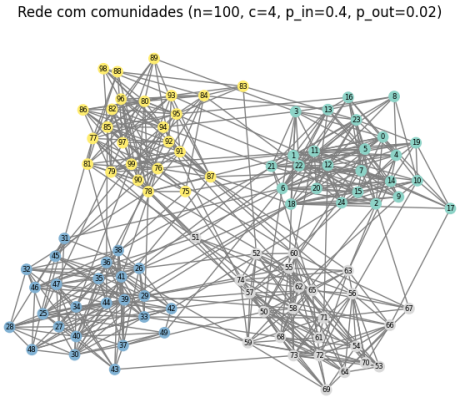


Figure 4: Rede gerada com comunidades ($p_{in} = 0{,}4$, $p_{out} = 0{,}02$).

Distribuição de Graus

A distribuição de graus apresenta maior concentração de conexões dentro dos grupos, enquanto nós de diferentes comunidades possuem poucos vínculos entre si.

Métricas da Rede

- **Número de componentes conexas:** 1
- **Diâmetro:** 4

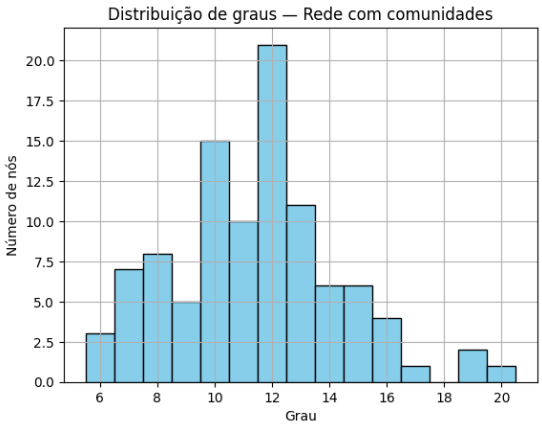


Figure 5: Distribuição de graus da rede gerada com comunidades.

- **Caminho médio:** 2.36
- **Grau médio:** 11.44
- **Desvio padrão dos graus:** 2.89
- **Coeficiente de clustering médio:** 0.31
- **Densidade:** 0.12

As métricas calculadas permitem caracterizar diferentes aspectos da rede. O **número de componentes conexas** indica se a rede está inteiramente conectada ou possui grupos isolados. O **diâmetro** corresponde à maior distância entre dois nós, enquanto o **caminho médio** representa a distância média entre todos os pares de nós, indicando o quão navegável é a rede. O **grau médio** mostra, em média, quantas conexões cada nó possui, e o **desvio padrão dos graus** revela a variação dessa quantidade, indicando se há nós muito mais conectados que outros. O **coeficiente de clustering médio** avalia a tendência de formação de triângulos na rede, ou seja, se os vizinhos de um nó também tendem a estar conectados entre si. Já a **densidade** mede o quão preenchida de arestas a rede está, considerando o número total de conexões possíveis.

1.3 Código Python das Redes com Comunidades

```
import networkx as nx
import matplotlib.pyplot as plt
import random
import numpy as np

def rede_comunidades(n=100, c=4, p_in=0.4, p_out=0.02):
```

```
"""
rede com comunidades:
- p_in: probabilidade de conexão dentro da comunidade
- p_out: probabilidade de conexão entre comunidades
"""
G = nx.Graph()
G.add_nodes_from(range(n))

# define as comunidades
tamanho = n // c
comunidade = {i: i // tamanho for i in range(n)}

# cria as arestas
for i in range(n):
    for j in range(i + 1, n):
        if comunidade[i] == comunidade[j]:
            if random.random() < p_in:
                G.add_edge(i, j)
        else:
            if random.random() < p_out:
                G.add_edge(i, j)

# visualização
cores = [comunidade[n] for n in G.nodes]
nx.draw(
    G,
    node_color=cores,
    cmap=plt.cm.Set3,
    with_labels=True,
    node_size=80,
    font_size=6,
    edge_color='gray'
)
plt.title(f"Rede com comunidades (n={n}, c={c}, p_in={p_in}, p_out={p_out})")
plt.show()

# métricas
graus = [grau for _, grau in G.degree()]

print(f"Número de componentes conexas: {nx.number_connected_components(G)}")
if nx.is_connected(G):
    print(f"Diâmetro: {nx.diameter(G)}")
    print(f"Caminho médio: {nx.average_shortest_path_length(G):.2f}")
else:
    print("Atenção: Grafo desconexo { não é possível calcular caminho médio e diâmetro.'
```

```
    print(f"Grau médio: {np.mean(graus):.2f}")
    print(f"Desvio padrão dos graus: {np.std(graus):.2f}")
    print(f"Coeficiente de clustering médio: {nx.average_clustering(G):.2f}")
    print(f"Densidade: {nx.density(G):.2f}")

    # histograma
    plt.hist(
        graus,
        bins=range(min(graus), max(graus) + 2),
        align='left',
        color='skyblue',
        edgecolor='black'
    )
    plt.title(f"Distribuição de graus | Rede com comunidades")
    plt.xlabel("Grau")
    plt.ylabel("Número de nós")
    plt.grid(True)
    plt.show()

    return G, comunidade

G, comunidade = rede_comunidades(n=100, c=4, p_in=0.4, p_out=0.02)
```

Conclusão

A rede gerada apresenta uma estrutura modular bem definida. O grau médio elevado reflete a alta conectividade dentro dos grupos, enquanto o clustering moderado (0.31) indica uma tendência à formação de triângulos, especialmente dentro das comunidades. O diâmetro (4) e o caminho médio (2.36) são relativamente baixos, indicando que, apesar da separação em grupos, a rede permanece bem navegável. A densidade (0.12) reforça que, embora haja boa conectividade interna, as conexões entre diferentes comunidades são escassas. As métricas refletem uma rede típica de sistemas com estrutura comunitária, como redes sociais, acadêmicas ou organizacionais.

Item 2.4 — Modelo de Crescimento: Anexação Uniforme

Metodologia

- **Rede inicial**: totalmente conectada com 10 nós.
- **Número de novos nós adicionados**: 5
- **Critério de ligação**:
  - Em um teste, cada novo nó estabeleceu **3 ligações fixas** com nós antigos.
  - Em outro teste, a quantidade de ligações foi **sorteada aleatoriamente** entre 1 e 4.
- **Implementação**: desenvolvida manualmente em `Python` com uso da biblioteca `networkx`.

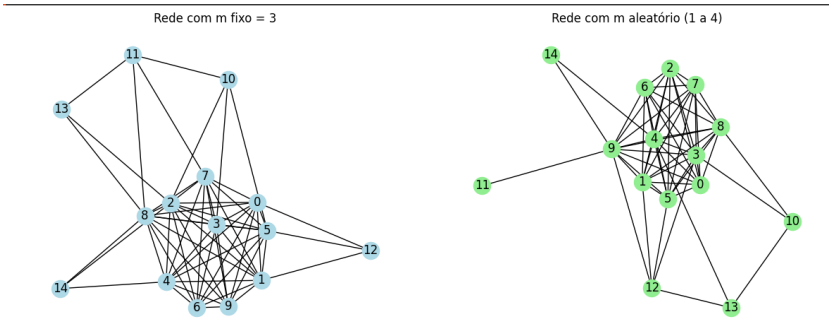
Diferença entre m fixo e m aleatório

Nos experimentos realizados, utilizamos duas abordagens distintas para determinar a quantidade de conexões que cada novo nó estabelece ao ser inserido na rede:

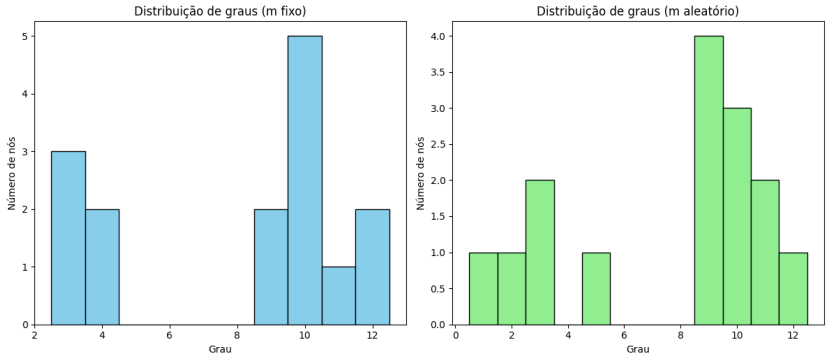
- ***m* fixo**: todos os novos nós vão realizar exatamente o mesmo número de conexões com nós já existentes. Essa configuração tende a gerar uma rede com distribuição de graus mais homogênea, na qual os nós apresentam graus parecidos entre si. No experimento, usamos $m = 3$.
- ***m* aleatório**: o número de conexões é sorteado individualmente para cada novo nó dentro de um intervalo pré-definido. Essa abordagem introduz variabilidade na estrutura da rede, o que vai resultar em uma distribuição de graus mais diferente e um maior desvio padrão. No experimento, sorteamos m aleatoriamente entre 1 e 4.

A comparação entre os dois métodos nos deixa observar como pequenas mudanças na regra de crescimento podem impactar grandemente as propriedades estruturais da rede, mesmo mantendo a lógica de anexação uniforme.

Visualização da Rede Final



Distribuição de Graus



As distribuições mostram a aleatoriedade do modelo. Observamos que os novos nós tendem a ter grau inferior aos nós iniciais. A distribuição não apresenta *hubs*, como é esperado em um modelo de anexação uniforme.

Métricas da Rede

As duas redes têm o mesmo número de nós, mas diferem um pouco no número de arestas, mostrando pequenas variações na conectividade. O grau médio representa o número médio de conexões por nó, que é alto em ambas as redes, sugerindo uma rede relativamente densa, o que também é confirmado pela densidade próxima a 0,5. O clustering médio mostra que existe uma tendência forte de os vizinhos estarem conectados entre si, o que indica a presença de grupos locais ou triângulos na rede. A única componente conexa confirma que a rede está totalmente conectada, sem ilhas isoladas. O desvio padrão dos graus revela variação moderada na distribuição dos graus dos nós. E por fim, o caminho médio e o diâmetro indicam que a distância entre os nós é pequena, o que caracteriza as redes pequenas e compactas.

Métrica	m fixo = 3	m aleatório (0-4)
Número de nós	15	15
Número de arestas	60	57
Grau médio	8.0	7.6
Clustering médio	0.7780	0.6997
Componentes conexas	1	1
Desvio padrão dos graus	3.37	3.57
Densidade	0.5714	0.5429
Caminho médio	1.46	1.49
Diâmetro	3	3

Table 1: Métricas calculadas para as duas redes analisadas

1.4 Código Python do Modelo De Anexação Uniforme

```
import networkx as nx
import matplotlib.pyplot as plt
import random
import numpy as np

def anexacao_uniforme(G, n_novos, m=None, aleatorio=False, min_m=1, max_m=3):
    """
    G com n_novos nós, conectando com anexação uniforme.

    parâmetros:
    - G: grafo inicial (NetworkX)
    - n_novos: número de novos nós a adicionar
    - m: número fixo de ligações por novo nó (ignorado se aleatorio=True)
    - aleatorio: se True, sorteia m entre min_m e max_m (inclusive)
    - min_m, max_m: limites de sorteio de m
    """
    proximo_id = max(G.nodes) + 1  # garante IDs únicos para novos nós

    for _ in range(n_novos):
        G.add_node(proximo_id)

        # define quantas conexões esse nó terá
        if aleatorio:
            m_atual = random.randint(min_m, min(max_m, len(G.nodes) - 1))
        else:
            m_atual = min(m, len(G.nodes) - 1)

        # escolhe aleatoriamente os nós existentes para conectar
        possiveis_alvos = list(G.nodes)
        possiveis_alvos.remove(proximo_id)  # não pode conectar com ele mesmo
        alvos = random.sample(possiveis_alvos, m_atual)
```

```
        for alvo in alvos:
            G.add_edge(proximo_id, alvo)

        proximo_id += 1

    return G

G_fixo = nx.complete_graph(10)
G_fixo = anexacao_uniforme(G_fixo, n_novos=5, m=3, aleatorio=False)

G_aleat = nx.complete_graph(10)
G_aleat = anexacao_uniforme(G_aleat, n_novos=5, aleatorio=True, min_m=1, max_m=4)

fig, axs = plt.subplots(2, 2, figsize=(12, 10))

# visualização das redes
nx.draw(G_fixo, with_labels=True, node_color='lightblue', ax=axs[0, 0])
axs[0, 0].set_title("Rede com m fixo = 3")

nx.draw(G_aleat, with_labels=True, node_color='lightgreen', ax=axs[0, 1])
axs[0, 1].set_title("Rede com m aleatório (1 a 4)")

# metricas
def imprimir_metricas(G):
    graus = [grau for _, grau in G.degree()]
    print("Número de nós:", G.number_of_nodes())
    print("Número de arestas:", G.number_of_edges())
    print("Grau médio:", np.mean(graus))
    print("Clustering médio:", nx.average_clustering(G))
    print("Componentes conexas:", nx.number_connected_components(G))
    print("Desvio padrão dos graus:", np.std(graus))
    print("Densidade:", nx.density(G))
    print("Caminho médio:", nx.average_shortest_path_length(G))
    print("Diâmetro:\n", nx.diameter(G))


imprimir_metricas(G_fixo)
imprimir_metricas(G_aleat)


# histogramas dos graus
graus_fixo = [grau for _, grau in G_fixo.degree()]
graus_aleat = [grau for _, grau in G_aleat.degree()]

axs[1, 0].hist(graus_fixo, bins=range(min(graus_fixo), max(graus_fixo)+2), align='left', col
```

```
axs[1, 0].set_title("Distribuição de graus (m fixo)")
axs[1, 0].set_xlabel("Grau")
axs[1, 0].set_ylabel("Número de nós")

axs[1, 1].hist(graus_aleat, bins=range(min(graus_aleat), max(graus_aleat)+2), align='left',
axs[1, 1].set_title("Distribuição de graus (m aleatório)")
axs[1, 1].set_xlabel("Grau")
axs[1, 1].set_ylabel("Número de nós")

plt.tight_layout()
plt.show()
```

Conclusão

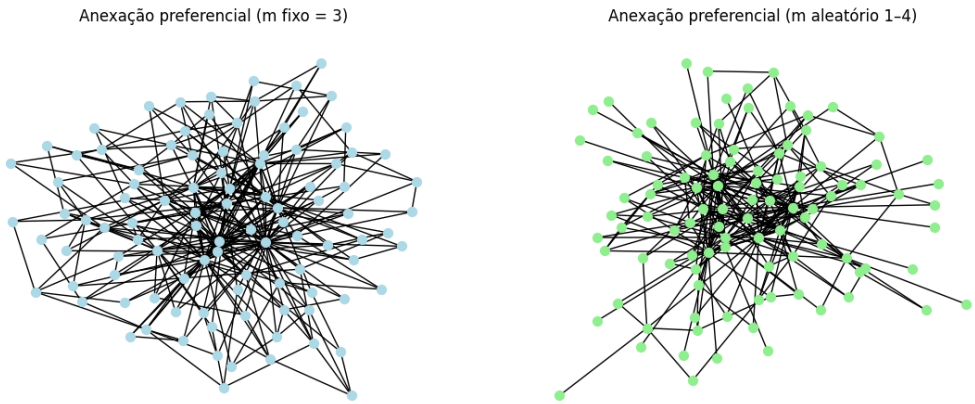
O modelo de anexação uniforme, como nós esperamos, gerou uma rede sem *hubs* e com distribuição de graus assimétrica. As métricas calculadas (como alto clustering e baixo caminho médio) mostram a influência da rede inicial totalmente conectada. Este comportamento fa um contraste com modelos baseados em anexação preferencial, onde a heterogeneidade de graus é mais destacada.

Item 2.5 — Modelo de Crescimento: Anexação Preferencial

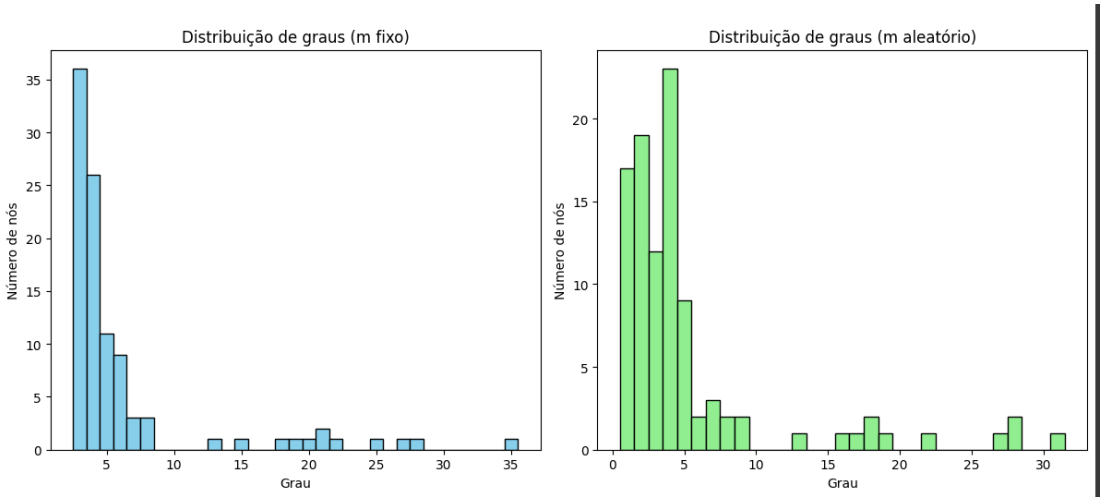
Metodologia

- **Rede inicial:** totalmente conectada com 10 nós.
- **Número de novos nós adicionados:** 90
- **Critério de ligação:**
  - Em um teste, cada novo nó estabeleceu 3 ligações fixas com nós antigos.
  - Em outro teste, a quantidade de ligações foi sorteada aleatoriamente entre 1 e 4.
- **Implementação:** desenvolvida manualmente em Python com uso da biblioteca `networkx` apenas para análise e visualização. A lógica de crescimento foi programada "na mão", de forma que cada novo nó seleciona os nós-alvo com probabilidade proporcional ao grau.

Visualização da Rede Final



Distribuição de Graus



A distribuição de graus da rede gerada por anexação preferencial apresenta **cauda longa**, com a maioria dos nós tendo poucos vizinhos e poucos nós concentrando um grande número de conexões (*hubs*). Esse comportamento é característico de redes *livre de escala* (*scale-free*).

Diferença entre m fixo e m aleatório

Nos dois testes realizados, observamos comportamentos distintos:

- m **fixo:** cada novo nó realizou exatamente 3 conexões. A presença de *hubs* ainda é evidente, pois o mecanismo de anexação preferencial privilegia nós que já possuem maior grau.
- m **aleatório:** variando entre 1 e 4 ligações, essa configuração introduz maior variabilidade na estrutura da rede. A distribuição de graus torna-se ainda mais dispersa, com maior desvio padrão.

Em ambos os casos, a estrutura resultante tende a se organizar com **nós altamente conectados (hubs)**, o que é uma característica marcante desse tipo de modelo.

Métricas da Rede

Métrica	**m fixo = 3**	**m aleatório (1–4)**
Número de nós	100	100
Número de arestas	315	271
Grau médio	6.3	5.42
Clustering médio	0.2930	0.2231
Componentes conexas	1	1
Desvio padrão dos graus	6.55	6.13
Densidade	0.0636	0.0547
Caminho médio	2.49	2.69
Diâmetro	5	6

Table 2: Métricas calculadas para as duas variações da rede

As duas redes possuem mesma quantidade de nós e uma única componente conexa, indicando que todos os nós estão interligados direta ou indiretamente. O número de arestas e o grau médio são relativamente baixos, o que é típico do modelo Barabási–Albert, que vai gerar redes esparsas. A densidade também é baixa, o que reforça essa característica. O clustering médio é moderado, o que é esperado nesse modelo, já que os novos nós se conectam preferencialmente a nós de alto grau, favorecendo a formação de hubs, mas não necessariamente triângulos. O desvio padrão dos graus é alto, refletindo a distribuição desigual de conexões, uma marca do modelo: poucos nós com grau alto (hubs) e muitos com grau baixo. Por fim, os caminhos médios curtos e os diâmetros pequenos evidenciam a propriedade de pequeno mundo, comum em redes com estrutura de hubs.

1.5 Código Python do Modelo de Anexação Preferencial

```
import networkx as nx
import matplotlib.pyplot as plt
import random
import numpy as np

def anexacao_preferencial(G, n_novos, m=None, aleatorio=False, min_m=1, max_m=3):
    """
    rede G com anexação preferencial (modelo Barabási-Albert).

    parâmetros:
    - G: grafo inicial (NetworkX)
    - n_novos: número de novos nós a adicionar
    - m: número fixo de ligações por novo nó (ignorado se aleatorio=True)
    - aleatorio: se True, sorteia m entre min_m e max_m (inclusive)
    - min_m, max_m: limites de sorteio de m
    """
    proximo_id = max(G.nodes) + 1

    for _ in range(n_novos):
        G.add_node(proximo_id)

        # define quantas conexões esse nó terá
        if aleatorio:
            m_atual = random.randint(min_m, min(max_m, len(G.nodes) - 1))
        else:
            m_atual = min(m, len(G.nodes) - 1)

        # calcular graus acumulados para anexação preferencial
        graus = np.array([G.degree(n) for n in G.nodes if n != proximo_id])
        nos_existentes = [n for n in G.nodes if n != proximo_id]

        if graus.sum() == 0:
            # todos com probabilidade igual se graus forem todos zero
            alvos = random.sample(nos_existentes, m_atual)
        else:
            probabilidades = graus / graus.sum()
            alvos = np.random.choice(nos_existentes, size=m_atual, replace=False, p=probabil

        for alvo in alvos:
            G.add_edge(proximo_id, alvo)

        proximo_id += 1

    return G
```

```
# grafo inicial completamente conectado
G_fixo = nx.complete_graph(10)
G_fixo = anexacao_preferencial(G_fixo, n_novos=90, m=3, aleatorio=False)

G_aleat = nx.complete_graph(10)
G_aleat = anexacao_preferencial(G_aleat, n_novos=90, aleatorio=True, min_m=1, max_m=4)

# metricas
def imprimir_metricas(G):
    graus = [grau for _, grau in G.degree()]
    print("Número de nós:", G.number_of_nodes())
    print("Número de arestas:", G.number_of_edges())
    print("Grau médio:", np.mean(graus))
    print("Clustering médio:", nx.average_clustering(G))
    print("Componentes conexas:", nx.number_connected_components(G))
    print("Desvio padrão dos graus:", np.std(graus))
    print("Densidade:", nx.density(G))
    print("Caminho médio:", nx.average_shortest_path_length(G))
    print("Diâmetro:\n", nx.diameter(G))


imprimir_metricas(G_fixo)
imprimir_metricas(G_aleat)


# visualização
fig, axs = plt.subplots(2, 2, figsize=(12, 10))

nx.draw(G_fixo, with_labels=False, node_color='lightblue', node_size=50, ax=axs[0, 0])
axs[0, 0].set_title("Anexação preferencial (m fixo = 3)")

nx.draw(G_aleat, with_labels=False, node_color='lightgreen', node_size=50, ax=axs[0, 1])
axs[0, 1].set_title("Anexação preferencial (m aleatório 1{4)")

graus_fixo = [grau for _, grau in G_fixo.degree()]
graus_aleat = [grau for _, grau in G_aleat.degree()]

axs[1, 0].hist(graus_fixo, bins=range(min(graus_fixo), max(graus_fixo)+2), align='left', col
axs[1, 0].set_title("Distribuição de graus (m fixo)")
axs[1, 0].set_xlabel("Grau")
axs[1, 0].set_ylabel("Número de nós")

axs[1, 1].hist(graus_aleat, bins=range(min(graus_aleat), max(graus_aleat)+2), align='left',
axs[1, 1].set_title("Distribuição de graus (m aleatório)")
axs[1, 1].set_xlabel("Grau")
```

```
axs[1, 1].set_ylabel("Número de nós")

plt.tight_layout()
plt.show()
```

Conclusão

O modelo de anexação preferencial resultou em redes com **hubs naturais**, onde poucos nós concentram grande parte das conexões. Isso causa um contraste forte com o modelo de anexação uniforme (item 2.4), no qual a distribuição de graus é mais homogênea.

Além disso, o clustering médio tende a ser menor, e o desvio padrão dos graus significativamente maior, o que mostra a **heterogeneidade estrutural** da rede. As propriedades são observadas em casos reais, como redes sociais e a internet.

Item 2.6 — Modelo de Crescimento: Modelo de Price

Metodologia

- **Rede inicial:** totalmente conectada com 10 nós.
- **Número de novos nós adicionados:** 90.
- **Critério de ligação:**
  - Em um teste, cada novo nó estabeleceu 3 ligações fixas com nós antigos ($m = 3$).
  - Em outro teste, o número de ligações foi sorteado aleatoriamente entre 1 e 4.
- **Parâmetro de controle:** 60% das ligações foram feitas via anexação preferencial, e 40% de forma uniforme (preferência = 0.6).
- **Implementação:** desenvolvida manualmente em Python com uso da biblioteca `networkx` apenas para análise e visualização. A lógica de crescimento foi programada diretamente.

Visualização da Rede Final

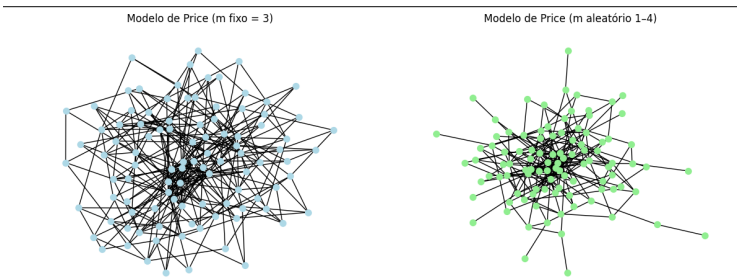


Figure 6: Redes geradas com o modelo de Price: à esquerda com $m = 3$, à direita com m sorteado entre 1 e 4.

Distribuição de Graus

As distribuições de graus mostram um padrão intermediário entre o modelo de anexação uniforme e o modelo preferencial puro. A presença de alguns hubs é visível, mas com menor intensidade do que no modelo de Barabási–Albert.

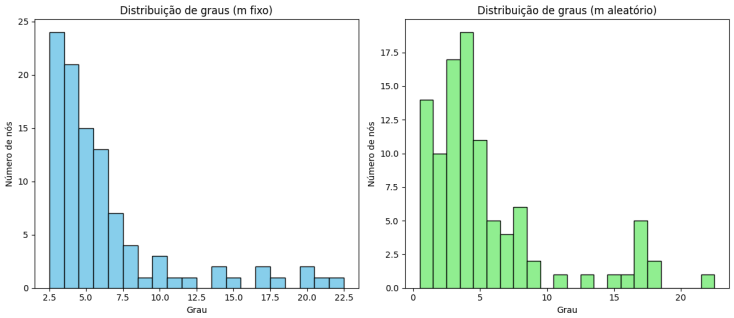


Figure 7: Distribuição de graus para o modelo de Price.

Diferença entre m fixo e m aleatório

Nos testes realizados, foi possível perceber os seguintes efeitos:

- **m fixo:** a rede resultante tende a apresentar uma estrutura mais regular, com menos variabilidade entre os graus dos nós. Os hubs surgem com menor destaque.
- **m aleatório:** há maior dispersão na distribuição de graus, com desvio padrão mais alto e maior variabilidade na estrutura.

Métricas da Rede

Métrica	**m fixo = 3**	**m aleatório (1–4)**
Número de nós	100	100
Número de arestas	315	269
Grau médio	6.30	5.38
Clustering médio	0.1030	0.1248
Componentes conexas	1	1
Desvio padrão dos graus	4.38	4.65
Densidade	0.0636	0.0543
Caminho médio	2.68	2.97
Diâmetro	5	7

Table 3: Métricas calculadas para as redes geradas com o modelo de Price.

As métricas obtidas mostram o comportamento misto do modelo de Price, combinando características de anexação uniforme e preferencial. Em ambos os testes, a rede final continua conectada (1 componente conexa), e o grau médio foi compatível com o número de ligações impostas durante o crescimento. A rede com **m fixo** resultou em um grau médio maior e um **diâmetro menor**, o que indica caminhos mais curtos entre os nós. Isso é esperado, já que todos os novos

nós mantêm uma quantidade constante de conexões, promovendo interligação mais regular. No cenário com **m aleatório**, a rede mostrou um **clustering médio pouco superior** e **maior diâmetro**, o que indica que a variabilidade na quantidade de conexões levou à formação de pequenas subestruturas, mas também aumentou as distâncias entre alguns pares de nós. O **desvio padrão dos graus** foi levemente maior no caso aleatório, o que nos confirma a maior heterogeneidade na distribuição de graus. Nenhum dos dois cenários apresenta clustering elevado, o que é coerente com a natureza não-local e parcialmente aleatória das conexões. Por fim, os resultados mostram que o parâmetro de anexação preferencial (60%) é suficiente para provocar o surgimento de alguns nós com grau significativamente superior (embora não tão marcante quanto no modelo de Barabási–Albert), o que acaba contribuindo para uma estrutura com características intermediárias entre aleatoriedade e formação de hubs.

1.6 Código Python do Modelo de Price

```
import networkx as nx
import matplotlib.pyplot as plt
import random
import numpy as np

def modelo_price(G, n_novos, m=None, aleatorio=False, min_m=1, max_m=3, preferencia=0.5):
    """
    rede G com o modelo de Price: parte preferencial, parte uniforme.

    parâmetros:
    - G: grafo inicial (NetworkX)
    - n_novos: número de novos nós a adicionar
    - m: número fixo de ligações por novo nó (ignorado se aleatorio=True)
    - aleatorio: se True, sorteia m entre min_m e max_m (inclusive)
    - preferencia: proporção das ligações feitas com anexação preferencial (entre 0 e 1)
    """
    proximo_id = max(G.nodes) + 1

    for _ in range(n_novos):
        G.add_node(proximo_id)

        # define quantas conexões esse nó terá
        if aleatorio:
            m_atual = random.randint(min_m, min(max_m, len(G.nodes) - 1))
        else:
            m_atual = min(m, len(G.nodes) - 1)

        # separa quantidade de conexões preferenciais e uniformes
        m_pref = int(preferencia * m_atual)
        m_uni = m_atual - m_pref
```

```
        # nós existentes, sem o novo nó
        nos_existentes = [n for n in G.nodes if n != proximo_id]

        # anexação preferencial
        alvos_pref = []
        if m_pref > 0:
            graus = np.array([G.degree(n) for n in nos_existentes])
            if graus.sum() == 0:
                alvos_pref = random.sample(nos_existentes, m_pref)
            else:
                probs = graus / graus.sum()
                alvos_pref = list(np.random.choice(nos_existentes, size=m_pref, replace=Fals

        # anexação uniforme
        alvos_restantes = list(set(nos_existentes) - set(alvos_pref))
        alvos_uni = random.sample(alvos_restantes, min(m_uni, len(alvos_restantes)))

        # adiciona as arestas
        for alvo in alvos_pref + alvos_uni:
            G.add_edge(proximo_id, alvo)

        proximo_id += 1

    return G

# redes iniciais
G_fixo = nx.complete_graph(10)
G_aleat = nx.complete_graph(10)

# modelo de Price
G_fixo = modelo_price(G_fixo, n_novos=90, m=3, preferencia=0.6, aleatorio=False)
G_aleat = modelo_price(G_aleat, n_novos=90, aleatorio=True, min_m=1, max_m=4, preferencia=0.

# visualização e análise
fig, axs = plt.subplots(2, 2, figsize=(12, 10))

nx.draw(G_fixo, with_labels=False, node_color='lightblue', node_size=50, ax=axs[0, 0])
axs[0, 0].set_title("Modelo de Price (m fixo = 3)")

nx.draw(G_aleat, with_labels=False, node_color='lightgreen', node_size=50, ax=axs[0, 1])
axs[0, 1].set_title("Modelo de Price (m aleatório 1{4)")

graus_fixo = [grau for _, grau in G_fixo.degree()]
graus_aleat = [grau for _, grau in G_aleat.degree()]
```

```
axs[1, 0].hist(graus_fixo, bins=range(min(graus_fixo), max(graus_fixo)+2), align='left', col
axs[1, 0].set_title("Distribuição de graus (m fixo)")
axs[1, 0].set_xlabel("Grau")
axs[1, 0].set_ylabel("Número de nós")

axs[1, 1].hist(graus_aleat, bins=range(min(graus_aleat), max(graus_aleat)+2), align='left',
axs[1, 1].set_title("Distribuição de graus (m aleatório)")
axs[1, 1].set_xlabel("Grau")
axs[1, 1].set_ylabel("Número de nós")

plt.tight_layout()
plt.show()

def imprimir_metricas(G):
    graus = [grau for _, grau in G.degree()]
    print("Número de nós:", G.number_of_nodes())
    print("Número de arestas:", G.number_of_edges())
    print("Grau médio:", np.mean(graus))
    print("Clustering médio:", nx.average_clustering(G))
    print("Componentes conexas:", nx.number_connected_components(G))
    print("Desvio padrão dos graus:", np.std(graus))
    print("Densidade:", nx.density(G))
    print("Caminho médio:", nx.average_shortest_path_length(G))
    print("Diâmetro:", nx.diameter(G))

imprimir_metricas(G_fixo)
imprimir_metricas(G_aleat)
```

Conclusão

O modelo de Price resultou em redes com propriedades intermediárias entre os modelos de anexação uniforme e preferencial. A presença parcial de anexação preferencial permitiu o surgimento de hubs moderados, enquanto a parte uniforme manteve certa homogeneidade na rede. As métricas refletem esse equilíbrio: a rede apresenta baixa densidade, grau médio moderado, clustering médio razoável e caminhos curtos, o que a caracteriza como uma rede com propriedades mistas. O experimento mostra como a simples variação do parâmetro de preferência pode influenciar significativamente a estrutura da rede.