مدل انتشار دوتایی SI روی شبکه‌ی ترکیبی باراباسی–آلبرت و دنیای کوچک

مطالعه‌ای جزئی بر فصل نه و ده کتاب dynamical processes on complex network داشتم و در حین انجام مراحل مختلف پروژه عمیق‌تر وارد مطالب کتاب می‌شوم. با مطالب فصل نه آشنایی بیشتری داشتم و هر دو فصل برایم بسیار جالب بودند. برای شروع شبکه‌ای که توصیف کرده بودید را ساخته و تا جای ممکن به مدلی که گفته بودید نزدیکش کردم.

شبکه‌ای داریم با 1000 رأس که اتصالات بین رئوس آن از دو نوع مختلف هستند. تعداد از رأس‌ها به روش باراباسی‌آلبرت به یک‌دیگر متصل بوده و دیگر رئوس به روش دنیای کوچک به هم‌دیگر متصل هستند. یال‌های گروه اول (باراباسی‌آلبرت) به رنگ قرمز و یال‌های گروه دوم(دنیای کوچک) به رنگ آبی می‌باشند. ابتدا ویژگی‌های این شبکه را توصیف می‌کنیم، سپس آن را شبیه‌سازی کرده و پخش اطلاعات درست و نادرست در آن را بررسی می‌کنیم.
تعداد رأس‌هایی که از روش باراباسی‌آلبرت پیروی می‌کنند N_r است. در زمان t با احتمال زیر رأسی انتخاب می‌شود:

$$\Pi_s(t) = \frac{k_s(t)}{\sum_j k_j(t)} \tag{۱}$$

سپس بین رأس انتخاب شده و رأس اضافه شده یک یال تعریف می‌شود. این کار m بار تکرار می‌شود تا رأس جدید دارای m یال باشد. آهنگ تغییر درجه رئوس قدیمی در هر گام زمانی T از رابطه زیر پیروی می‌کند:

$$\frac{\mathrm{d}k_s}{\mathrm{d}t} = m\Pi_s(t) = \frac{mk_s(t)}{2mt + m_0 \langle k \rangle_0} \tag{۲}$$

در $t \to \infty$ پاسخ معادله بالا $k_s(t) \approx m(\frac{t}{s})^{1/2}$ می‌باشد. همچنین تابع توزیع درجه رأس برابر است با:

$$P_{BA}(k,t) = \frac{2m^2}{t+m_0}k^{-3} \xrightarrow{t\to\infty} P_{BA}(k) = 2m^2k^{-3} \tag{۳}$$

حال گروه دیگر را بررسی می‌کنیم. تعداد رأس‌هایی که اتصال بین آن‌ها به صورت دنیای کوچک است را N_b در نظر می‌گیریم. تعداد N_b رأس را در یک حلقه قرار می‌دیم و هر رأس را به $m2$ همسایه‌اش، m از راست و m از چپ، متصل می‌کنیم. یک یال (i,j) را به تصادف انتخاب کرده و با احتمال p این یال را حذف می‌کنیم. به طور یکنواخت یک رأس w دیگر انتخاب می‌کنیم، به طوری که خود i نباشد و جز رئوسی که به i متصل هستند نباشد،و یک اتصال جدید بین i و w تعریف می‌کنیم. امکان تعریف چند یال بین دو رأس یا یال از یک رأس به خودش وجود ندارد! این روش میانگین درجه را بر روی $\langle k \rangle = 2m$ نگه می‌دارد و تابع توزیع درجه رأس به این صورت است:

$$P_{SW}(k) = \sum^{\min(k-m,m)} \binom{m}{n}(1-p)^n p^{m-n}\frac{(pm)^{k-m-n}}{(k-m-n)!}$$

در واقع پروسه ساختن شبکه را به این صورت در نظر گرفتم که با N_r رأس شبکه باراباسی آلبرت و با N_b شبکه دنیای

کوچک خود را می‌سازیم و در آخر صرفا این دو را میکس می‌کنیم. ترجیح می‌دادم از یک روشی استفاده کنم که هر رأس بتواند هر دو یال آبی و قرمز را داشته باشد! به این صورت که در ساخت یال‌های آبی و قرمز رأس‌های مشترکی وجود داشته باشد. اما در این حالت مشکلی که پیش می‌آمد، امکان وجود دو یال آبی و قرمز بین دو رأس مشخص بود. به دلیل پیچیدگی‌های زیاد ابتدا با این روش ساده‌تر پیش می‌روم شاید بعدها آن‌را پیچیده‌تر کردم!

با استفاده از پایتون شبکه توصیف شده را ساختم و متن کد پایتون در زیر قرار داده شده است:

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

N_total   = 1000      # total vertices
N_BA      = 300       # vertices grown by BA (IDs 0 … N_BA-1)
avg_k     = 6         #  mean degree → m = avg_k / 2 must be an integer
p_rewire  = 0.10      # WS rewiring probability
seed      = 42
random.seed(seed)
m = avg_k // 2        # m = 3 if avg_k = 6


red_nodes  = list(range(N_BA))            # BA vertices
blue_nodes = list(range(N_BA, N_total))   # WS vertices

# BUILD -BARABASIALBERT (red)
edges_red = []                      # list of (u, v)
deg       = {v: 0 for v in red_nodes}

# (a) fully-connected seed of size m+1
seed_clique = red_nodes[: m + 1]
for i, u in enumerate(seed_clique):
for v in seed_clique[i+1:]:
edges_red.append((u, v))
deg[u] += 1; deg[v] += 1

# (b) preferential-attachment growth
for new in red_nodes[m + 1:]:

targets = set()
while len(targets) < m:
r, cum, total = random.uniform(0, sum(deg.values())), 0, 0
for node, k in deg.items():
cum += k
if cum >= r: targets.add(node); break
for t in targets:
edges_red.append((new, t))
deg[new] = deg.get(new, 0) + 1
deg[t]  += 1

# BUILD -WATTSSTROGATZ (blue)
edges_blue = []
B = len(blue_nodes)
def gid(i):
return blue_nodes[i]

# (a) regular ring (degree 2m)
```

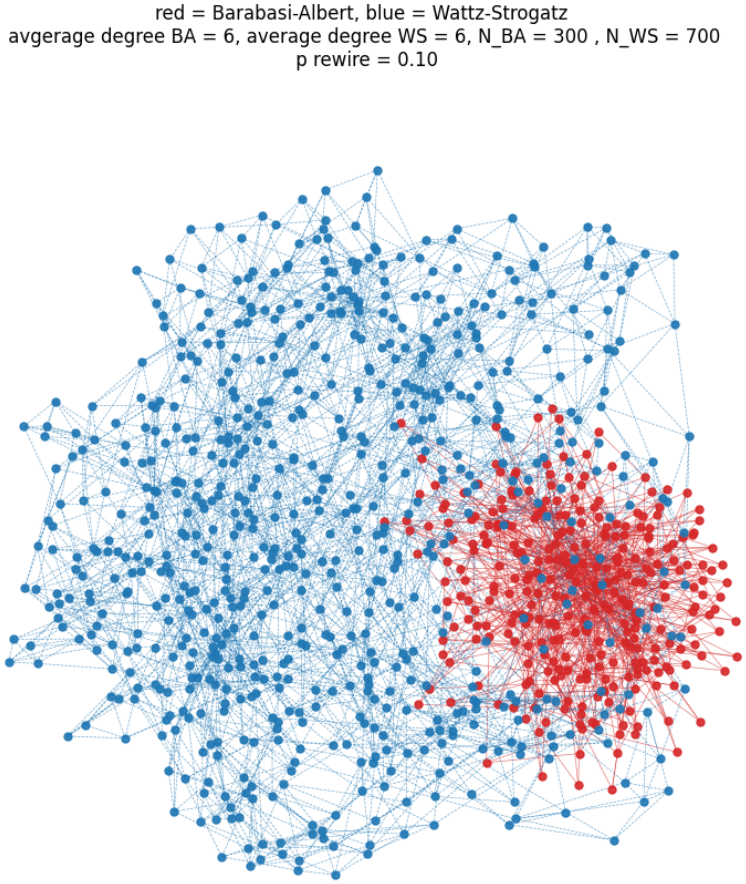
```
for i in range(B):
for d in range(1, m + 1):
j = (i + d) % B                    # clockwise neighbour
edges_blue.append((gid(i), gid(j)))

# (b) single-pass rewiring (clockwise edges only!)
for i in range(B):
for d in range(1, m + 1):
j = (i + d) % B
u, v = gid(i), gid(j)
if random.random() < p_rewire:
while True:                        # pick new endpoint w
w = gid(random.randrange(B))
if w != u and (u, w) not in edges_blue and (w, u) not in edges_blue:
break
edges_blue.remove((u, v))          # delete old lattice edge
edges_blue.append((u, w))          # add new

# BUILD COMBINED GRAPH
G = nx.Graph()
G.add_nodes_from(range(N_total))
G.add_edges_from(edges_red + edges_blue)  pos = nx.spring_layout(G, seed=
  seed, k=0.15)
```

برای استفاده از مدل باراباسی‌آلبرت باید یک شبکه اولیه داشته باشیم و که در آن درجه حداقل چند تا از رئوس مساوی با صفر نیست. بخش fully-connected seed of size m+۱ یک شبکه اولیه می‌سازد. سپس بخش growth اعمال می‌شود که رئوس را تک تک اضافه کرده و یال‌ها را تعریف می‌کند. سپس در بخش بعدی بخش دنیای کوچک ساخته می‌شود. در پارت (a) یک حلقه ساده ساخته شده و بعد یال‌ها به طور رندوم جابه‌جا می‌شوند و ویژگی دنیای کوچک به آن اضافه می‌شود. در آخر این دو بخش به هم اضافه می‌شوند.

عکس گراف این شبکه برای میانگین درجه 6، $N_{BA} = 300$ ، احتمال جابه‌جایی یال در گروه دوم 0.1 در زیر قرار داده شده‌است.



شکل ۱:

تغییر مقادیر پارامترها در این مرحله تقاوت زیادی ایجاد نمی‌کند و صرفا شکل کلی شبکه را عوض می‌کند. در مراحل بعدی که پخش شایعه را بررسی می‌کنیم اما این مقادیر مهم‌تر خواهند بود و حالت‌های مختلف را تحلیل خواهیم کرد.

ابتدا کل شبکه را در حالت S قرار می‌دهیم. سپس به طور رندوم دو جایگاه انتخاب می‌کنیم و یکی را در حالت T یا درست قرار می‌دهیم و دیگری را در حالت F یا نادرست. سپس با پارامترهای مختلف اجازه می‌دهیم این دو در شبکه پخش بشوند. در آخر می‌خواهیم ببینیم چند درصد از خبر درست و چند درصد خبر نادرست را شنیده‌اند. از آن‌جایی که دو بخش شبکه که با دو مدل ساخته‌ایم اتصالی با یکدیگر ندارند، برای اینکه پخش را در شبکه به درستی بررسی کنیم، از هر بخش به طور تصادفی دو رأس را انتخاب کرده و فرآیند را بر روی آن‌ها اعمال می کنیم. همچنین در فرآیند مورد بررسی ما هنگامی که یک رأس یک خبر درست یا نادرست را بشنود و آن را باور کند(به آن دجار شود) دیگر نظر او تغییر نخواهد کرد.
پارامترها برای دینامیک پخش T و F به این صورت تعریف می‌شوند:

- N: تعداد کل رأس‌ها
- β: ضریب آهنگ شیوع
- $I(t)$: تعداد رأس‌های آلوده شده یا تحت تاثیر قرار گرفته در زمان t

- B: تعداد رأس‌هایی که تحت تاثیر گرفته‌اند و به حالت مخالف حالت مورد بررسی دچار شده‌اند.
- $S(t) = N - B - I(t)$ تعداد رأس‌هایی که در حالت S هستند و آماده تحت تاثیر قرار گرفتن هستند.

برای یک رأس در حالت اولیه S با درجه k و تعداد همسایه مبتلای n، احتمال مبتلا نشدن در زمان $\mathrm{d}t$ برابر است با $(1-\beta\,\mathrm{d}t)^n$. بنابراین احتمال مبتلا شدن آن برابر است با:

$$\begin{aligned} &P(\text{vertex (i) getting infected}) = 1-(1-\beta\,\mathrm{d}t)^n \\ &\xrightarrow{\beta \mathrm{d}t \ll 1} P(\text{vertex (i) getting infected}) \approx \beta n\,\mathrm{d}t \end{aligned} \tag{۴}$$

حال با استفاده از تقریب میدان میانگین داریم:

$$i(t) = \frac{I(t)}{N} \implies P_{inf}(k) = \beta k i(t)\,\mathrm{d}t \tag{۵}$$

$$\Delta I(t) = \beta \frac{I(t)\,\langle k\rangle}{N} S(t)\,\mathrm{d}t \xrightarrow{\Delta t \to \mathrm{d}t} \frac{\mathrm{d}I}{\mathrm{d}t} = \beta \frac{\langle k\rangle}{N} IS \tag{۶}$$

$$\frac{\mathrm{d}i}{\mathrm{d}t} = \beta\,\langle k\rangle\, i \frac{S}{N} = \beta\,\langle k\rangle\, i \left(1 - \underbrace{\frac{B}{N}}_{b} - i\right) = \beta\,\langle k\rangle\, i(1-i-b) \tag{۷}$$

با در نظر گرفتن شرط اولیه $i(0) = i_0$ می‌توان معادله بالا را حل کرد.

$$\lambda \equiv \beta\langle k\rangle \to \frac{\mathrm{d}i}{i(1-b-i)} = \lambda\ \mathrm{d}t \tag{۸}$$

$$\begin{aligned} &\frac{1}{\rho(1-b-i)} = \frac{1}{1-b}\left(\frac{1}{i} + \frac{1}{1-b-i}\right) \\ &\xrightarrow{(۸)} \int \frac{1}{i(1-b-i)}\ \mathrm{d}i = \frac{1}{1-b}\int\left(\frac{1}{i} + \frac{1}{1-b-i}\right)\mathrm{d}i = \frac{1}{1-b}\left[\ln(i) - \ln(1-b-i)\right] \\ &\frac{1}{1-b}\left[\ln(i) - \ln(1-b-i)\right] = \lambda\, t + C \end{aligned} \tag{۹}$$

$$\begin{aligned} &\xrightarrow{t=0} C = \frac{1}{1-b}\left[\ln(i_0) - \ln(1-b-i_0)\right] \\ &\implies \boxed{i(t) = \frac{i_0\, e^{\lambda t}}{1-b-i_0+i_0\, e^{\lambda t}}} \end{aligned} \tag{۱۰}$$

حال دینامیک کلی شبکه را بررسی می‌کنیم. شبکه دارای دو لایه باراباسی‌آلبرت و دنیای کوچک است. هر لایه با $l \in \{r, b\}$ نشان داده می‌شود. خبر درست با آهنگ سرایت β_t و خبر نادرست با آهنگ β_f. میانگین درجه برای لایه r (باراباسی‌آلبرت) برابر $\langle k\rangle_r$ و برای لایه b (دنیای کوچک) برابر با $\langle k\rangle_b$ می‌باشد. در این مرحله فرض می‌کنیم که آهنگ انتشار خبر درست و نادرست در کل شبکه یکسان است. در مراحل بعدی می‌توانیم احتمال پخش خبر درست در یکی از لایه‌ها را کمتر در

نظر بگیریم.

$$\text{for each layer :} \quad \begin{cases} \dot{T}_\ell = \lambda_\ell^t T_\ell S_\ell \quad ; \quad \lambda_\ell^t = \beta_t \langle k \rangle_\ell \\ \dot{F}_\ell = \lambda_\ell^f F_e l l S_\ell \quad ; \quad \lambda_\ell^f = \beta_f \langle k \rangle_\ell \\ S_\ell = 1 - T_\ell - F_\ell \implies \dot{S} = -\left(\lambda_\ell^t + \lambda_\ell^f F\right) S \end{cases} \tag{۱۱}$$

در ابتدای فرآیند کسر کسانی که در حالت S هستند حدودا برابر یک است. بنابراین نسبت $R(t) = \frac{T(t)}{F(t)}$ بسیار آرام تغییر می‌کند. داریم:

$$T(t) = \frac{R_0}{1+R_0} I(t) \qquad , \qquad F(t) = \frac{1}{1+R_0} I(t) \tag{۱۲}$$

$$\dot{I} = \dot{T} + \dot{F} = \left(\lambda_\ell^t T_\ell + \lambda_\ell^f F_\ell\right) S \tag{۱۳}$$

رابطه (۱۲) و $S = 1 - I$ را جاگذاری می‌کنیم:

$$\dot{I} = \Lambda_\ell I(1-I) \quad ; \quad \lambda_\ell = \lambda_\ell^t \frac{R_0}{1+R_0} + \lambda_\ell^f \frac{1}{1+R_0} \tag{۱۴}$$

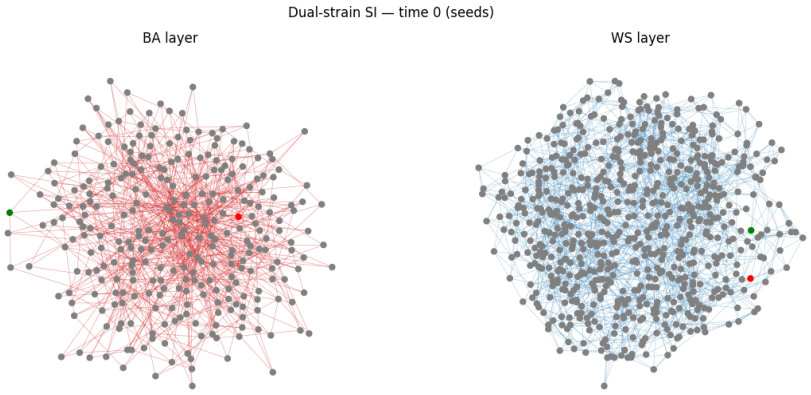
$$\int_{I_0}^{I(t)} \frac{dI}{I(1-I)} = \Lambda_\ell \int_0^t dt \to \Big[\ln I - \ln(1-I)\Big]_{I_0}^{I(t)} = \Lambda_\ell t$$

$$\ln \frac{I(t)}{1-I(t)} - \ln \frac{I_0}{1-I_0} = \Lambda_\ell t \implies \frac{I(t)}{1-I(t)} = \frac{I_0}{1-I_0}\, e^{\Lambda_\ell t} \tag{۱۵}$$

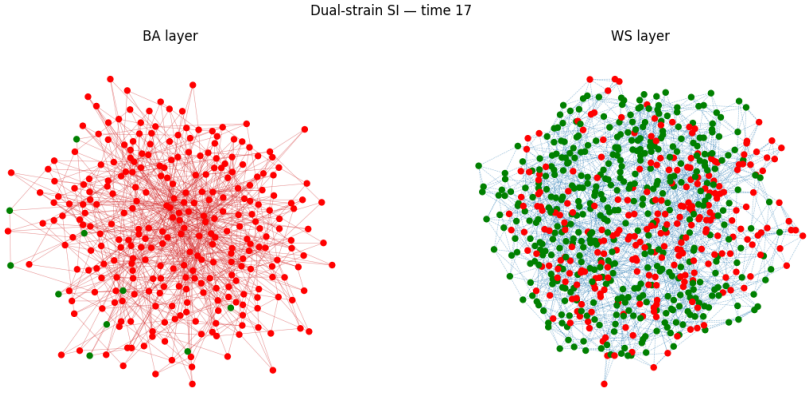
$$I(t) = \frac{I_0\, e^{\Lambda_\ell t}}{1 - I_0 + I_0\, e^{\Lambda_\ell t}} \tag{۱۶}$$

حال شبیه‌سازی شبکه را با پایتون انجام می‌دهیم. ابتدا آهنگ پخش خبر درست و نادرست را در دو لایه مساوی قرار می‌دهیم. عکس از حالت اولیه و نهایی برای دو اجرای کد در زیر قرار گرفته است.

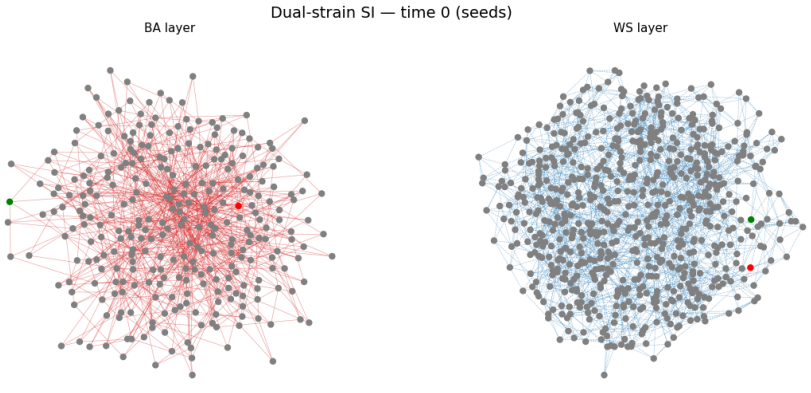
$$\beta = 0.3$$



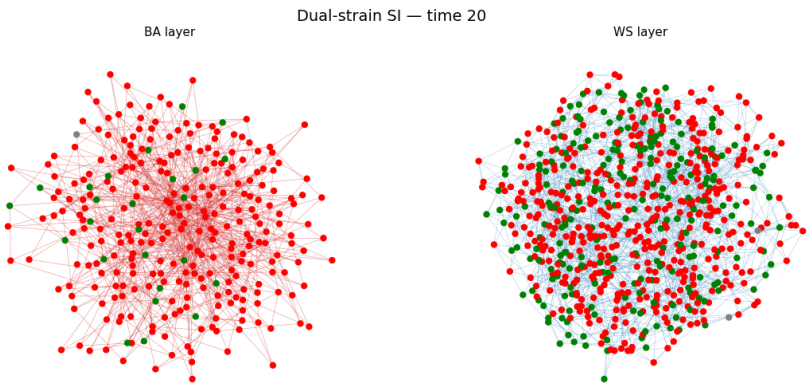
شکل ۲:



شکل ۳:



شکل ۴:



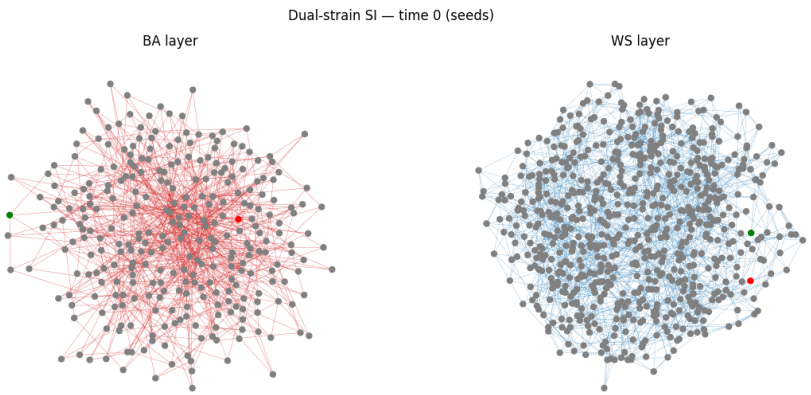
شکل ۵:

در هر دو اجرای کد، در لایه باراباسی‌آلبرت به تصادف رأسی با درجه بیشتری برای F انتخاب شده و به همین دلیل انتشار آن در شبکه بسیار بیشتر است. اما در لایه دنیای کوچک انتشار F و T تقریبا به صورت متوازن بوده و در آخر تقریبا نصف قرمز و نصف سبز هستند. اگر بارها دیگر کد را اجرا کنیم،متوجه می‌شویم که در لایه باراباسی‌آلبرت، هر کدام از T و F که در ابتدا رأس با درجه بیشتری برایش انتخاب شود، در آخر درصد زیادی از این لایه را خواهد پوشاندو اما از طرف دیگر در لایه دنیای کوچک، انتشار هر دو خبر تقریبا یکسان بوده زیرا درجه رأس‌ها به صورت یکنواخت پخش شده‌اند و احتمال اینکه F یا T بر اساس انتخاب اولیه پخش بیشتری داشته باشند کمتر است. درصدهای T و F در اولین اجرا به صورت زیر بوده:

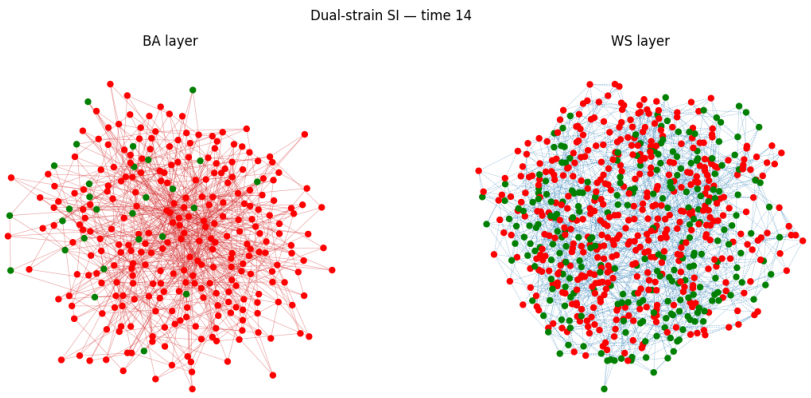
- Global: T = 44.60% , F = 55.40%
- BA layer: T = 3.33% , F = 96.67%
- WS layer: T = 62.29% , F = 37.71%

حال β را برای F و T متفاوت انتخاب می‌کنیم.

$$\beta_T = 0.3 \quad , \quad \beta_F = 0.4$$



شکل ۶:



شکل ۷:

- Global: T = 27.80% , F = 72.20%
- BA layer: T = 9.33% , F = 90.67%
- WS layer: T = 35.71% , F = 64.29%

در این حالت که β_F بیشتر است در هر دو لایه قرمز یا F غالب است که با چیزی که انتظار داشتیم منطبق می‌باشد.
کد این بخش نیز در زیر قرار گرفته است.

```
beta_T = 0.30
beta_F = 0.40
rng    = random.Random(2024)

#  SEED STATES
nx.set_node_attributes(G, 'S',    'state')    # susceptible
nx.set_node_attributes(G, 'grey', 'colour')

def pick_two(group):
a = rng.choice(group);  b = rng.choice([v for v in group if v != a])
return a, b

Tred, Fred   = pick_two(red_nodes)
Tblue, Fblue = pick_two(blue_nodes)

for v in (Tred, Tblue):                  # strain-T seeds
G.nodes[v]['state'] = 'IT'; G.nodes[v]['colour'] = 'green'
for v in (Fred, Fblue):                  # strain-F seeds
G.nodes[v]['state'] = 'IF'; G.nodes[v]['colour'] = 'red'

#  UPDATE RULE
def step_dual_SI(G):
frontier = [n for n,d in G.nodes(data=True) if d['state'] in ('IT','IF')]
rng.shuffle(frontier)                                    # random order

for u in frontier:
```

```
st_u = G.nodes[u]['state']
beta = beta_T if st_u == 'IT' else beta_F

for v in G.neighbors(u):
if G.nodes[v]['state'] == 'S' and rng.random() < beta:
# infect once; cannot flip later
if st_u == 'IT':
G.nodes[v]['state']  = 'IT'
G.nodes[v]['colour'] = 'green'
else:
G.nodes[v]['state']  = 'IF'
G.nodes[v]['colour'] = 'red'

#  TWO-PANEL DRAWING
pos_red  = nx.spring_layout(G.subgraph(red_nodes),  seed=42, k=0.15)
pos_blue = nx.spring_layout(G.subgraph(blue_nodes), seed=24, k=0.15)

def draw_layers(t_text="time 0"):
col = nx.get_node_attributes(G, 'colour'); default = 'grey'
fig, ax = plt.subplots(1, 2, figsize=(14, 6))

nx.draw_networkx_edges(G.subgraph(red_nodes),  pos_red,  ax=ax[0],
edge_color='tab:red',  width=0.4, alpha=0.5)
nx.draw_networkx_nodes(G.subgraph(red_nodes),  pos_red,  ax=ax[0],
node_color=[col.get(n, default) for n in red_nodes],
node_size=25)
ax[0].set_title("BA layer"); ax[0].axis('off')

nx.draw_networkx_edges(G.subgraph(blue_nodes), pos_blue, ax=ax[1],
edge_color='tab:blue', width=0.4, alpha=0.5, style='dashed')
nx.draw_networkx_nodes(G.subgraph(blue_nodes), pos_blue, ax=ax[1],
node_color=[col.get(n, default) for n in blue_nodes],
node_size=25)
ax[1].set_title("WS layer"); ax[1].axis('off')

fig.suptitle(f"Dual-strain SI - {t_text}", y=0.97)
plt.show()

draw_layers("time 0 (seeds)")

for t in range(1, 21):
step_dual_SI(G)
draw_layers(f"time {t}")
if all(G.nodes[v]['state'] != 'S' for v in G.nodes):
break
```