به نام خدا

دانشگاه صنعتی امیرکبیر

Amirkabir University
of Technology

تمرین اول درس تحلیل شبکه های پیچیده

GNP, Influence Maximization, Outbreak Detection

استاد درس: دکترچهرقانی

نام: زهرا اخلاقی

شماره دانشجویی: ۴۰۱۱۳۱۰۶۴

پاییز ۱۴۰۲

فهرست مطالب

سوال اول: بررسی ویژگی‌های گرافی 2
الف) 2
ب) 2
ج) 2
د) 3
ه) 9
و) 12
سوال دوم: مفاهیم گراف تصادفی 14
الف) 14
ب) 15
ج) 16
سوال سوم: تشخیص شیوع 17
الف) 17
ب) 17
ج) 18
د) 20
ه) 21
و) 22
سوال چهارم: بیشینه‌سازی تاثیر 24
الف) 24
ب) 24
ج) 25
د) 26
ه) 26
و) 27

سوال اول: بررسی ویژگی‌های گرافی

الف)

گراف انتخابی، graph1 است و نتایج به صورت زیر است:

Number of nodes: 3718

Number of edges: 91723

ب)

```python
def generate_erdos_renyi_graph(num_nodes, num_edges, p):
    graph = {node: [] for node in range(num_nodes)}
    count = 0

    while count < num_nodes:
        for i in range(num_nodes):
            for j in range(i + 1, num_nodes):
                if random.random() < p and count < num_edges and j not in graph[i]:
                    graph[i].append(j)
                    graph[j].append(i)
                    count += 1

    return graph
```

تابع بالا گراف Erdos-renyi را تولید میگیرد و تعداد گره، یال و احتمال را به عنوان ورودی می‌گیرد. در این تابع به تعداد یال های ورودی یال تولید می‌شود و برای تولید هر یال احتمالی در نظر گرفته شده است (مقدار احتمال در هر بار اجرا شدن این تابع برابر با ۰.۲ در نظر گرفته شده است).

ج)

در تابع زیر برای تولید گراف small world، تعداد گره، یال و احتمال را به عنوان ورودی می‌گیرد. در این تابع هر گره با همسایگان خود یال ایجاد میکند و در فاز reward هر یال با احتمال ۰.۲ به یکی دیگر از گرهها متصل می‌شود.

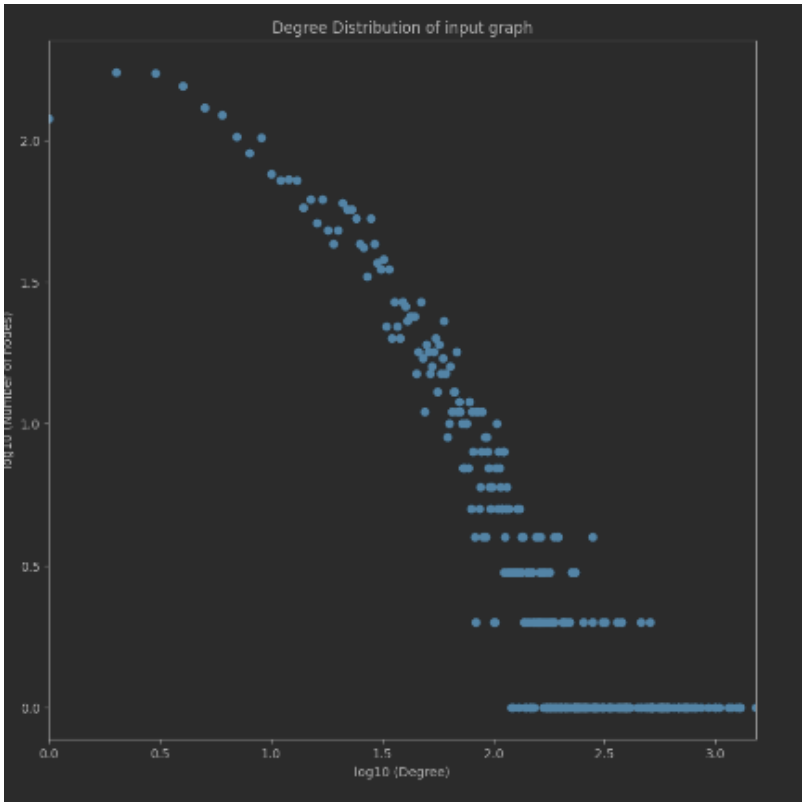
در نهایت به علت اینکه تعداد یال‌ها باید با تعداد یال ورودی باشد، اگر تعداد یال تولید شده کمتر بود یال تصادفی ایجاد می‌کند.

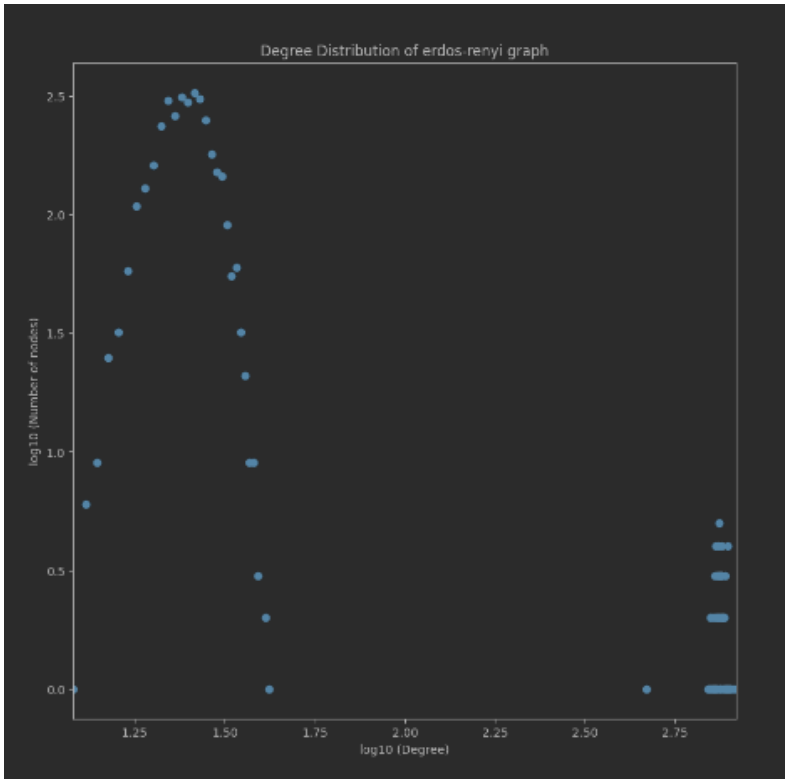
```python
def generate_small_world_graph(num_vertices, num_edges, rewiring_probability):
    count = 0
    if num_edges < num_vertices:
        raise ValueError("The number of edges should be greater than or equal to the number of vertices.")

    k = num_edges // num_vertices
    graph = {node: [] for node in range(num_vertices)}

    for i in range(num_vertices):
        for j in range(0, k):
            neighbor = (i + j) % num_vertices
            if neighbor not in graph[i] and i not in graph[neighbor] and i != neighbor:
                graph[i].append(neighbor)
                graph[neighbor].append(i)
                count += 1

    for i in graph.keys():
        for j in graph[i]:
            if random.random() < rewiring_probability:
                new_neighbor = random.randint(0, num_vertices - 1)
                if new_neighbor != i:
                    graph[i].remove(j)
                    graph[j].remove(i)
                    graph[i].append(new_neighbor)
                    graph[new_neighbor].append(i)

    diff = num_edges - count
    while diff > 0:
        new_neighbor1 = random.randint(0, num_vertices - 1)
        new_neighbor2 = random.randint(0, num_vertices - 1)
        if new_neighbor2 not in graph[new_neighbor1]:
            graph[new_neighbor2].append(new_neighbor1)
            graph[new_neighbor1].append(new_neighbor2)

            count += 1
            diff -= 1
    return graph
```

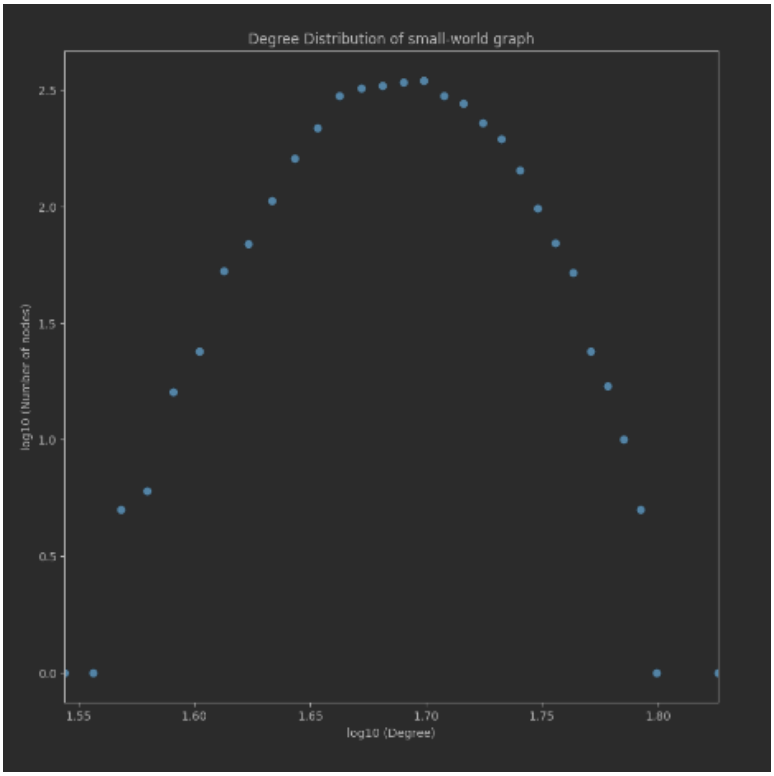
د)

نمودار توزیع درجه سه گراف پیوست شده و حالت Erdos-Renyi و small-world هر سه گراف پیوست شده، در ادامه ارائه شده است.

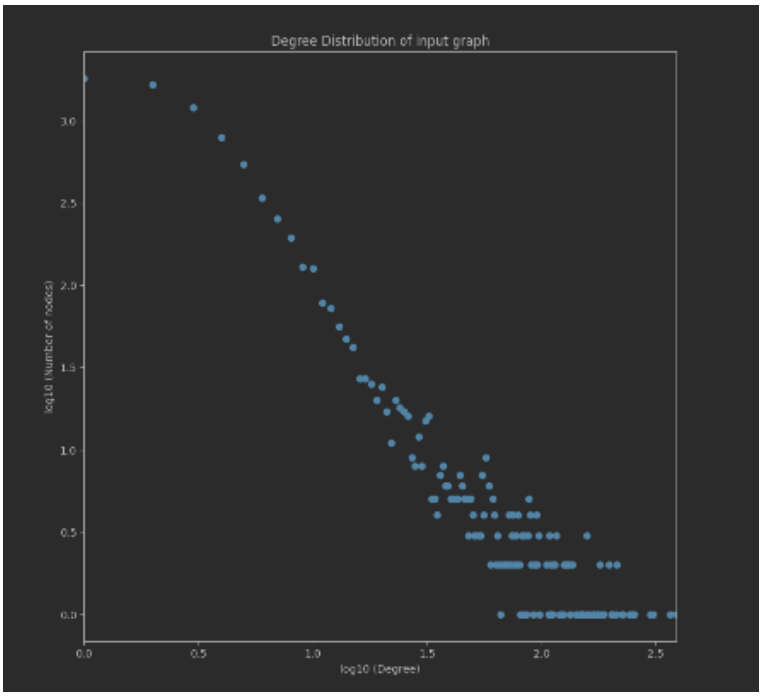
گراف اول:

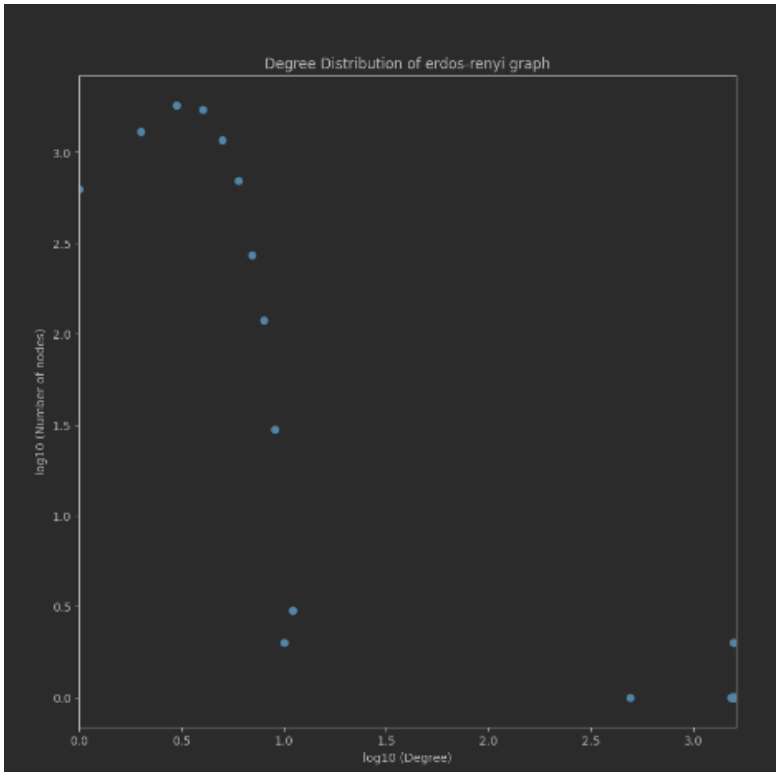

Degree Distribution of input graph
log10 (Number of nodes)
log10 (Degree)
0.0
0.5
1.0
1.5
2.0
2.5
3.0

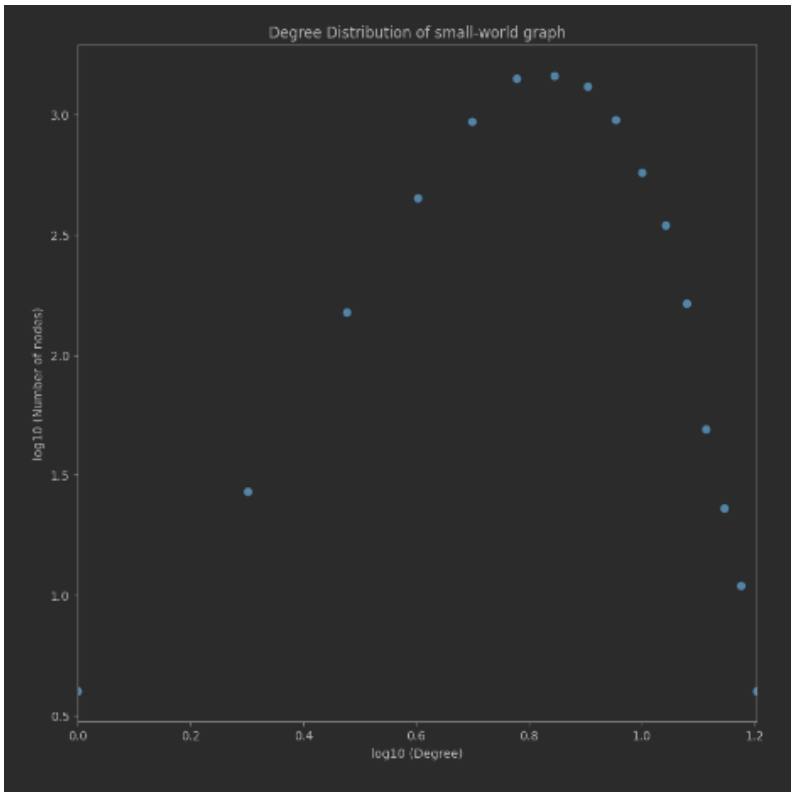

Degree Distribution of erdos renyi graph
log10 (Number of nodes)
log10 (Degree)
0.0
0.5
1.0
1.5
2.0
2.5
1.25
1.50
1.75
2.00
2.25
2.50
2.75


Degree Distribution of small-world graph
log10 (Number of nodes)
log10 (Degree)
2.5
2.0
1.5
1.0
0.5
0.0
1.55
1.60
1.65
1.70
1.75
1.80

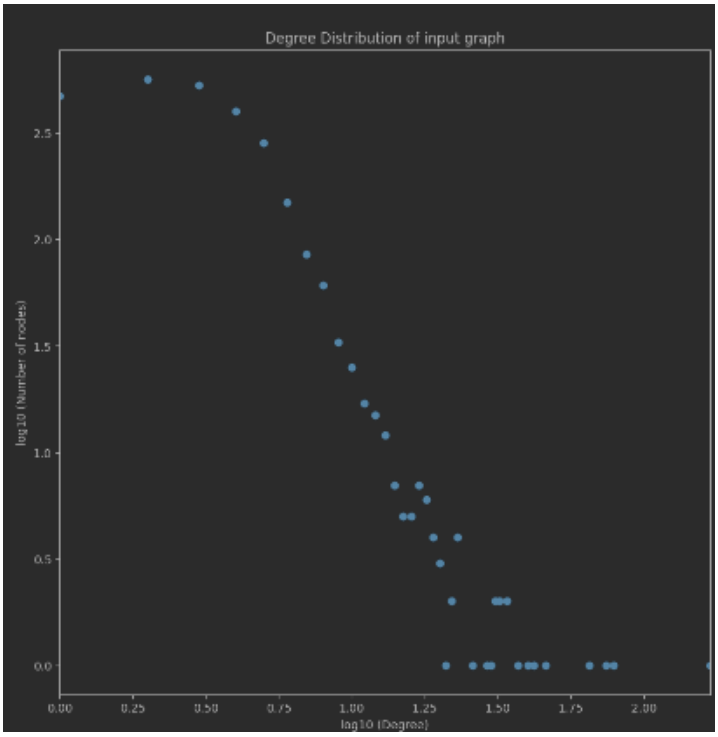
گراف دوم:

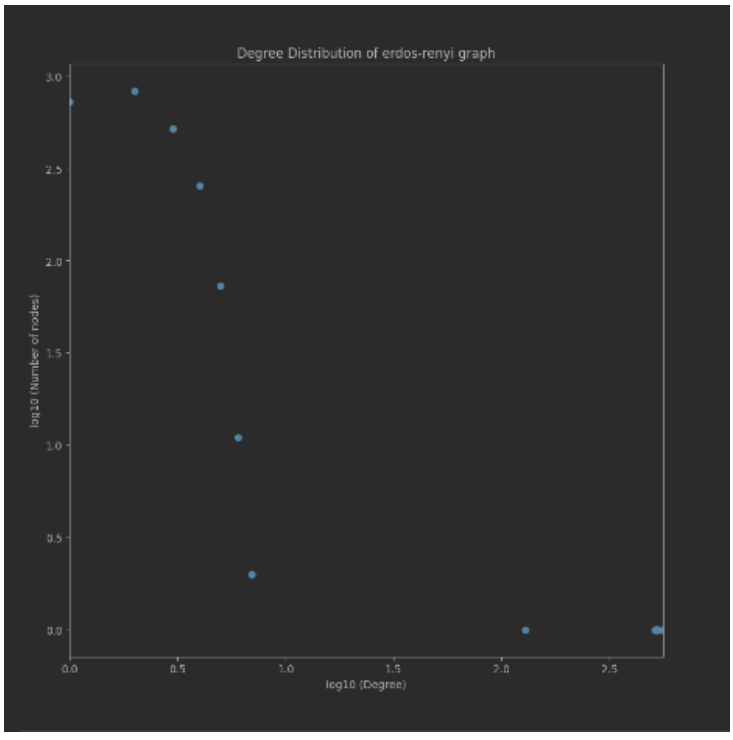

Degree Distribution of input graph
log10 (Number of nodes)
log10 (Degree)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5

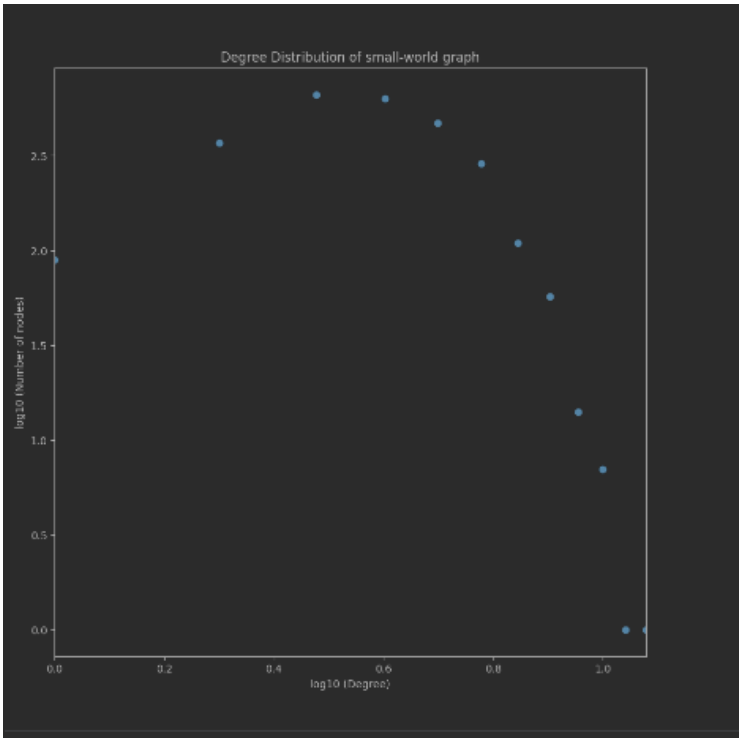

Degree Distribution of erdos-renyi graph
log10 (Number of nodes)
log10 (Degree)
0.0
0.5
1.0
1.5
2.0
2.5
3.0


Degree Distribution of small-world graph
log10 (Number of nodes)
log10 (Degree)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0.5
1.0
1.5
2.0
2.5
3.0

گراف سوم:


Degree Distribution of input graph
log10 (Number of nodes)
log10 (Degree)
0.0
0.5
1.0
1.5
2.0
2.5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Degree Distribution of erdos-renyi graph
log10 (Number of nodes)
log10 (Degree)
0.0
0.5
1.0
1.5
2.0
2.5
3.0



توزیع درجه در گراف Erdos-Renyi از توزیع پوآسونی پیروی می‌کند. این به این معنا که اغلب گره‌ها درجه کمی دارند و تعداد گره‌هایی که درجه بالایی دارند نادر هستند. این توزیع به صورت آماری تقریبی به توزیع پوآسونی نزدیک می‌شود.

توزیع درجه در گراف Small-World به طور معمول نسبت به Erdos-Renyi متغیرتر و پویا تر است. به این معنا که در این گونه از گراف، بسیاری از گره‌ها درجه کمی دارند ولی همچنان تعداد گره‌هایی با درجه بالا نیز وجود دارد. این تنوع در توزیع درجه به دلیل وجود کوتاه‌ترین مسیرهای میان گره‌ها و تشکیل گروه‌های کوچک و بزرگ از گره‌ها ناشی می‌شود. توزیع درجه در گراف Erdos-Renyi به شکل پوآسونی متمرکز بر اتصالات کمتر است، در حالی که توزیع درجه در گراف Small-World متغیرتر است.

در گراف small-world در ابتدا همه گره‌ها درجه یکسانی دارند و تنها با همسایگان خود یال دارند و در مرحله بعد به صورت تصادفی با احتمالی با گره‌های دیگر یال ایجاد می کند که باعث ایجاد تنوع در توزیع درجه می‌شود.

گراف‌های دنیای واقعی تفاوت‌های بسیاری با گراف‌های مدلی مانند Erdos-Renyi یا Small-World دارند. این تفاوت‌ها به دلیل خصوصیات و ساختارهای مختلفی است که در گراف‌های دنیای واقعی وجود دارد. برخی از این تفاوت‌ها عبارتند از:

1. توزیع درجه: گراف‌های دنیای واقعی عمدتاً توزیع درجه‌های پویا و متغیر دارند. در بسیاری از شبکه‌های واقعی، تعداد کمی گره دارای درجه بالا (مراکز) و تعداد زیادی گره دارای درجه کم (گره‌های عمومی) وجود دارد.
2. ساختار اجتماعی: گراف‌های دنیای واقعی اغلب دارای ساختارهای اجتماعی مشخص هستند که در مدل‌های ساده‌تر مانند Erdos-Renyi و Small-World نمی‌یابیم. این ساختارهای اجتماعی معمولاً به دلیل رفتارهای انسانی، تعامل‌های اجتماعی و شبکه‌های واقعی میان افراد یا موجودات به وجود می‌آیند.
3. دینامیک و تکامل: گراف‌های دنیای واقعی تغییرات زمانی و تکاملی دارند. این به معنای آن است که اتصالات بین گره‌ها ممکن است در طول زمان تغییر کنند و گراف به شکل پویا تکامل یابد.

بنابراین، تفاوت‌ها بین گراف‌های دنیای واقعی و مدل‌های ساده‌تر از نظر توزیع درجه، ساختار اجتماعی، دینامیک و نوع شبکه می‌تواند بسیار مهم و متنوع باشد و باید با توجه به موضوع مورد بررسی در نظر گرفته شود.
تفاوت‌های میان گراف‌های دنیای واقعی به شدت وابسته به نوع شبکه، منشأ داده، اندازه، ویژگی‌ها و ساختار اجتماعی هر گراف می‌باشد. در سه شکل حاصل از گراف دنیای واقعی اگرچه ممکن است سه شکل به یکدیگر شبیه باشند ولی مقیاس آنها در محور x,y یکدیگر متفاوت است و در هر سه گراف احتمال وجود گره با درجه بزرگ صفر نیست و توزیع درجه به صورت power-low میباشد و در گراف اول و دوم گره‌هایی با درجه بزرگ بیشتر وجود دارند.

ه)

مسیر متوسط یک گراف، میانگین طول همهٔ مسیرهای ممکن بین دو گره مختلف است و قطر گراف، بزرگترین طول میان همهٔ مسیرهای ممکن بین دو گره در گراف است. در ادامه طول مسیر متوسط و قطر گراف برای هر سه گراف پیوست شده در حالت بی‌جهت و جهت دار ارائه شده است. میانگین طول مسیر در گراف بی‌جهت در giant componrnt آن گزارش شده و در گراف جهت دار strongly connected component گراف برای محاسبه مسیر متوسط استفاده شده است.
تابع kosaraju_scc برای محاسبه strongl connected component در گراف جهت‌دار و تابع find_giant_component برای محاسبه giant component در گراف بی‌جهت استفاده شده است.

گراف اول:

Undirected Graph:

average path length:2.66217 diameter:6

در این گراف اندازه SCC=1 است بنابراین تمام گره‌ها برای محاسبه avg path length , diameter استفاده شده است و نتیجه به صورت زیر است:

Directed Graph:

average path length:0.00664 diameter:1

گراف دوم:

Undirected Graph:

average path length:3.99304 diameter:8

در این گراف اندازه SCC=1 است بنابراین تمام گره‌ها برای محاسبه avg path length , diameter استفاده شده است و نتیجه به صورت زیر است:

Directed Graph:

average path length:0.00046 diameter:1

گراف سوم:

Undirected Graph:

average path length:6.31100 diameter:19

در این گراف اندازه SCC=27 است و گره‌های SCC برای محاسبه avg path length , diameter استفاده شده است و نتیجه به صورت زیر است:

Directed Graph:

average path length:2.82051 diameter:11

در هر سه گراف طول مسیر متوسط و قطر گراف در حالت بی‌جهت بزرگتر از جهتدار است که ممکن است به علت زیر باشد:

- اتصالات بین گره‌ها در گراف بدون‌جهت به صورت دو طرفه می‌باشند. این ویژگی تعداد کمی از مسیرهای بین گره‌ها را کوتاهتر می‌کند، اما در عین حال ممکن است مسیرهای دیگر را بلندتر کند. این تفاوت‌ها در تعداد مسیرها و اتصالات دو طرفه باعث ایجاد تفاوت در طول مسیر متوسط و قطر گراف می‌شوند.
- در گراف جهت‌دار، ممکن است ویژگی‌های خاصی در توپولوژی گراف وجود داشته باشد که باعث کوتاه‌تر شدن برخی از مسیرها و افزایش کمتری در طول مسیر متوسط شود. به عبارت دیگر، وجود جهت‌ها می‌تواند برخی مسیرها را به حداقل برساند. قطر گراف ممکن است بزرگتر باشد چرا که جهت‌ها ممکن است مانع از وجود مسیرهای مستقیم بین برخی از گره‌ها شوند و مسیرهای جایگزین بلندتری ایجاد کنند.

در محاسبه طول مسیر میانگین با توجه به اینکه در رابطه مجموع فاصله کوتاه ترین فاصله بین دو راس بر تعداد یال گراف کامل ممکن در giant component تقسیم میشود، هر چه تعداد رئوس در giant component یک گراف بیشتر باشد، بر عدد بزرگتری تقسیم شده و میانگین طول مسیر عدد کمتری به دست میاید، در گراف اول و دوم کل گراف یک giant component میباشد و چون تعداد مسیرها بر عدد بزرگ تقسیم شده است طول مسیر میانگین برابر با صفر است.

قطر گراف به عوامل متعددی بستگی دارد. برخی از عوامل مهم عبارتند از:

- تعداد گره‌ها: با افزایش تعداد گره‌ها، احتمال بزرگ شدن قطر گراف نیز افزایش می‌یابد. گراف‌های بزرگتر معمولاً قطر بزرگتری دارند.
- نوع اتصالات: در گراف‌هایی که اتصالات بیشتری بین گره‌ها وجود دارد، ممکن است قطر گراف کمتری داشته باشد.
- ساختار گراف: ویژگی‌های خاص گراف مانند وجود گره‌های مهم یا تعداد کامپونت‌ها و اتصالات آنها.

طول مسیر متوسط در گراف به تعداد گره‌ها، نوع اتصالات و ساختار گراف بستگی دارد.

در گراف بی‌جهت، اتصالات بین گره‌ها دوطرفه هستند و مسیر بین هر زوج گره در هر دو جهت امکان‌پذیر است و بیانگر میانگین طول تمام مسیرهای ممکن بین هر زوج گره است و در گراف های فوق در حالت بی جهت طول مسیر میانگین بزرگتر است زیرا مسیر در هر دو جهت امکان پذیر است.

در گراف جهتدار، اتصالات به صورت جهت‌دار هستند و مسیر بین گره‌ها در یک جهت مشخص می‌شود و ممکن است امکان رسیدن به گره دیگر در حالت جهت دار نسبت به بیجهت اصلا وجود نداشته باشد یا مسیر طولانی‌تر باشد.

و)

در ادامه Clustering Coefficient سه گراف پیوست شده و حالت Erdos-Renyi و small-world هر سه گراف پیوست شده، در ادامه ارائه شده است.

گراف اول:

Average Clustering Coefficient of Input Graph: 0.0

Average Clustering Coefficient of Erdos_Renyi Graph: 0.20067033512647015

Average Clustering Coefficient of Small_World Graph: 0.6163544873464555

گراف دوم:

Average Clustering Coefficient of Input Graph: 0.0

Average Clustering Coefficient of Erdos_Renyi Graph: 0.19937491873242355

Average Clustering Coefficient of Small_World Graph: 0.3673865590754754

گراف سوم:

Average Clustering Coefficient of Input Graph: 0.23772863418575124

Average Clustering Coefficient of Erdos_Renyi Graph: 0.18405811278153952

Average Clustering Coefficient of Small_World Graph: 0.1852381689355699

در گراف small-world مقدار clustering coefficient بزرگتر از گراف تصادفی است زیرا برای ایجاد یالها در گراف small-world ابتدا هر گره با همسایگان خود یال ایجاد می کند و سپس با احتمالی ممکن است این یالها تغییر کنند که باعث میشود این گراف در ساختار خود ارتباط گره‌های نزدیک به هم را داشته باشد ولی در گراف تصادفی کاملا رندوم یالها انتخاب میشوند که ممکن است گره های همسایه را شامل نشود به این معناست که در گراف‌های تصادفی، اتصالات بین گره‌ها به صورت تصادفی ایجاد می‌شوند و ممکن است خوشه‌ها به نسبت کمتری وجود داشته باشند.

در گراف اول و دوم مقدار clustering coefficient برابر با صفر باشد، این به معنی عدم وجود خوشه‌بندی در

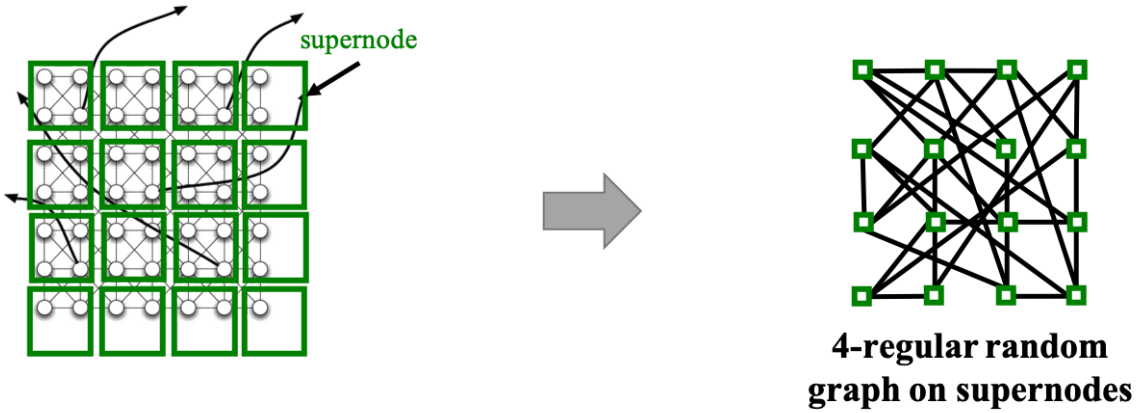
گراف است. به عبارت دیگر، هیچ گونه اتصالات میان گره‌های همسایه در گراف وجود ندارد و هیچ خوشه‌ای تشکیل نشده است. این وضعیت به عنوان یک گراف خطی یا گرافی با ساختاری بسیار ساده توصیف می‌شود در یک گراف خطی (یا گرافی بدون خوشه‌بندی)، هر گره تنها با دو گره دیگر ارتباط دارد و هیچ ارتباط بین همسایه‌های یک گره وجود ندارد. این به این معناست که هیچ گونه ارتباط مثلثی (تشکیل خوشه) در گراف وجود ندارد. این نوع گراف ممکن است به عنوان یک گراف خطی یا گرافی با ساختار شبکه‌ای بسیار ساده توصیف شود و در واقعیت، بیشتر از یک خط یا زنجیر به نظر می‌رسد.

این نوع گراف‌ها معمولاً در مواردی که ارتباطات میان اشیاء یا گره‌ها بسیار محدود و خاص به یکدیگر باشند، ظاهر می‌شوند. به عنوان مثال، در شبکه‌هایی که ارتباطات نادر هستند و ارتباطات تنها برخاسته از شرایط خاصی باشند (مانند ارتباطات حضوری بین اشیاء)، ممکن است گراف‌های بدون خوشه‌بندی ظاهر شوند. این نوع گراف‌ها معمولاً از خواص ساده‌تری برخوردارند و در مواردی که تنوع و پیچیدگی اتصالات پایین است، قابل مشاهده هستند.

سوال دوم: مفاهیم گراف تصادفی

الف)

برای به دست آوردن قطر در گراف Watts-Strogatz مدل ساده‌تری را بررسی میکنیم، شبکه ای که در آن زیرگراف های 2x2 را به عنوان یک ابرگره در نظر میگیریم، یک نمودار تصادفی 4 منظم ایجاد می کنیم (در ابتدا یک grid دو بعدی میسازیم و برای هر گره یک نیم یال ایجاد میکنیم و نیم یالها را به صورت تصادفی جفت میکنیم) بنابراین هر ابرگره ۴ یال دارد.



هر ۴ گره را یک ابرگره در نظر میگیریم و بین هر دو ابر گره یک یال ایجاد می‌شود اگر و فقط اگر بین یکی از گره‌های ابرگره اول و یکی از گره‌های ابرگره دوم، یک یال تصادفی وجود داشته باشد (یالهای تصادفی را ignore میکنیم). بنابراین گراف حاصل یک گراف منظم تصادفی است.

مانند گراف تصادفی Erdos-Renyi ، یک گراف تصادفی با اندازه n دارای قطر log^{n} است، که در آن n تعداد ابرگره ها است. علاوه بر این، از آنجایی که در هر ابرگره، مسیر حداکثر ممکن است یک گام دیگر پرش کند، در نتیجه تعداد گام‌ها برای تکمیل یک مسیر در مقایسه با تنها در نظر گرفتن ابرگره‌ها حداکثر دو برابر می‌شود. در نتیجه، قطر گراف Watts-Strogatz برابر با $O(2\ log^{n})$ می شود.

ب)

Sj را مجموعه ای از تمام گره های موجود در نظر میگیریم که در j مراحل BFS با شروع از s به آنها میرسیم.

برای اثبات Sj و Sj+1 را به هم مرتبط می‌کنیم.

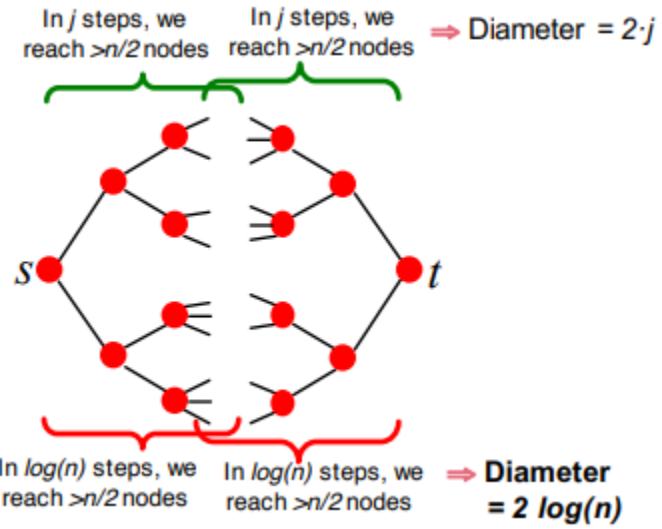
$$|S_{j+1}| = |S_j| + \text{node neighbor to } S_j$$

$$\geq |S_j| + \frac{\text{edge leaving } S_j}{\text{max edge arriving to a node}}$$

$$\geq |S_j| + \frac{\text{edge leaving } S_j}{k}$$

$$\geq |S_j| + \frac{\alpha|S_j|}{k}$$

$$|S_{j+1}| \geq |S_j|\left(1 + \frac{\alpha}{k}\right) = \left(1 + \frac{\alpha}{k}\right)^{j+1}$$



زمانی که $j = \frac{k \log_2 n}{\alpha}$ به بیشتر از n/2 گره در BFS میرسیم یعنی: $|S_j| = \left(1 + \frac{\alpha}{k}\right)^j \geq \frac{n}{2}$.

در نتیجه، داریم: $\left(1 + \frac{\alpha}{k}\right)^{\frac{k \log_2 n}{\alpha}} \geq 2^{\log_2 n} = n \geq \frac{n}{2}$

اثبات رابطه بالا:

$$\text{If } \alpha = k: \ (1 + 1)^{\frac{1 \log_2 n}{1}} = 2^{\log_2 n}$$

$$\text{If } \alpha \to 0 \text{ then } \frac{k}{\alpha} = x \to \infty: \text{ and } \left(1 + \frac{1}{x}\right)^{\frac{x \log_2 n}{1}} = e^{\log_2 n} \geq 2^{\log_2 n}$$

بنابراین در 2k/α·log n مرحله |Sj | به Θ(n) می رسد. بنابراین، قطر G O(log(n)/α) است.

ج)

Clustering coefficient از رابطه زیر محاسبه میشود:

$$Ci = \frac{2e_i}{ki\,(ki-1)}$$

در این رابطه ei تعداد یالها بین همسایگان گره i و ki برابر با تعداد همسایگان گره i می‌باشد.

در گراف Gnp هر یال با احتمال p رخ می‌دهد و ei یک random variable است که به جای آن expected value را قرار می‌دهیم و داریم:

$$e_i = p\,\frac{ki(ki-1)}{2}$$

با جایگذاری ei در فرمول clustering coefficient داریم:

$$Ci = \frac{p\,.\,ki\,(ki-1)}{ki\,(ki-1)} = p$$

به علت اینکه همه Ci ها با یکدیگر برابرند متوسط آنها نیز همین مقدار را دارد:

$$C = P = \frac{\bar{k}}{n-1} \simeq \frac{\bar{k}}{n} = \frac{avg\ degree}{n}$$

سوال سوم: تشخیص شیوع

الف)

Monotonic: if $S \subseteq T$ then $f(S) \leq f(T)$ and $f(\{\})=0$

زمانی که عضو جدیدی به مجموعه اضافه میشود تابع f برای آن بزرگتر شود و یا تغییری نکند و هیچگاه کمتر نشود.

submodular: $\forall S \subseteq T$ then $f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T)$

اگر عضو جدیدی به زیر مجموعه یک مجموعه اضافه شود gain بیشتری نسبت به زمانی که به خود مجموعه اضافه شود به دست میاورد.

اگر f را تابعی در نظر بگیریم که تعداد زیرمجموعه‌های یک مجموعه را حساب می‌کند، به علت اینکه تعداد زیر مجموعه های یک مجموعه برابر 2 به توان n است، با افزودن عضو جدید به مجموعه تابع f برای آن بزرگتر میشود و هیچگاه کمتر نمی شود، بنابراین خاصیت monotonic را دارد.

If n> m then 2^n > 2^m

خاصیت submodular را ندارد، مثال نقض:

S={a,b} => f(S) = 2^2 =4 T={a,b,c} => f(T) = 2^3=8

با افزودن عضو جدید {u} به مجموعه S , T داریم:

$f(S \cup \{u\}) = 2^3 =8$ $f(T \cup \{u\}) = 2^4=16$

بنابراین داریم:

$f(S \cup \{u\}) - f(S) = 8-4=4$ $f(T \cup \{u\}) - f(T) = 16-8=8$

بنابراین رابطه $f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T$ برقرار نیست.

ب)

از مشکلات الگوریتم حریصانه کندبودن و حجم بالای محاسبات است که در الگوریتم CELF تا حدودی این مشکلات وجود ندارد. در الگوریتم CELF در مرحله اول برای همه‌ی گره‌ها gain محاسبه شده و در لیست sort می‌کنیم و در هرمرحله لیست مرتبی از gain گره‌ها در مرحله قبل داریم که نزولی است و برای انتخاب هر گره، گره بالایی لیست را برداشته و gain آن را محاسبه می‌کنیم و با گره بعدی لیست مقایسه می‌کنیم و اگر gain گره برداشته‌شده از گره بعدی بزرگتر بود مناسب است و برای دیگر گره ها نیازی به محاسبه gain نداریم و اگر

این شرایط برقرار نبود این عمل را با گره بعدی تکرار می‌کنیم. الگوریتم CELF از ویژگی submodular استفاده می کند، که به این معنی است که گسترش گره‌ها در یک تکرار از الگوریتم Greedy نمی تواند بزرگتر از گسترش آن در تکرار قبلی باشد.نوآوری‌های الگوریتم CELF:

1. در الگوریتم CELF، از یک رویکرد تنبیهی یا lazy evaluation برای محاسبه تاثیر گره‌ها استفاده می‌شود. این به این معنی است که الگوریتم به طور معمول از پیش محاسباتی انجام نمی‌دهد و تنها محاسبات مربوط به تاثیر گره‌ها زمانی انجام می‌شود که این گره‌ها به مجموعه انتخابی اضافه می‌شوند. به عبارت دیگر، CELF تنها محاسباتی را انجام می‌دهد که به واقع برای تصمیم‌گیری ضروری هستند و این امر به بهبود کارایی الگوریتم منجر می‌شود.
2. الگوریتم CELF در مسئله تشخیص شیوع دو روش بدون در نظر گرفتن هزینه و انتخای بر اساس benefit/cost را اجرا میکند و هر کدام نتیجه بهتری داشتند (مقدار بیشتری برای f(S)) به عنوان نتیجه نهایی ارائه میدهد.

ج)

الگوریتم حریصانه به دو روش امکان پذیر است و برای گراف‌ها با ساختار متفاوت ممکن است یکی از این دو روش کارایی بهتری داشته باشد. منظور از gain تعداد گره هایی است که با انتخاب آن گره تحت تاثیر قرار میگیرند (تعدا سالن‌هایی که امکان گرفتن موش در آن گره‌ها وجود دارد).

الگوریتم حریصانه بدون در نظر گرفتن هزینه:

در این الگوریتم gain همه‌ی گره ها محاسبه شده و گره‌ای با بیشترین gain انتخاب میشود و هزینه‌ی آن از بودجه کم میشود و این روند ادامه پیدا میکند تا با محدودیت بودجه امکان انتخاب گره جدید نباشد.

node	gain	cost
A	۱	۳۰۰
B	۱	٤۰۰
C	۵	۵۰۰
D	۲	۵۰۰
E	۱	۲۰۰
F	۱	۳۰۰
G	۱	۲۰۰
H	۲	۵۰۰
I	۳	۸۰۰
J	٦	۸۰۰
K	۱	۷۰۰
L	۲	۷۰۰
M	۲	۹۰۰
N	٤	٤۰۰

با توجه به جدول بالا گره J دارای بیشترین مقدار gain میباشد و انتخاب میشود و هزینه باقی مانده ۲۰۰ هزار تومان است. در مرحله بعد gain نقاط برای تک تک گره ها دوباره محاسبه می‌شود (gain حاصل از اجتماع گره با J محاسبه شده است):

node	gain	cost	gain after selected J
A	۱	۳۰۰	۷
B	۱	٤۰۰	۷
C	۵	۵۰۰	۱۱
D	۲	۵۰۰	۸
E	۱	۲۰۰	۷
F	۱	۳۰۰	٦
G	۱	۲۰۰	٦
H	۲	۵۰۰	٦
I	۳	۸۰۰	٦
J	٦	۸۰۰	
K	۱	۷۰۰	٦
L	۲	۷۰۰	۷
M	۲	۹۰۰	۷
N	٤	٤۰۰	۸

با توجه به محدودیت هزینه تنها امکان انتخاب گره E , G وجود دارد و براساس جدول بالا gain نقطه E بیشتر است، بنابراین انتخاب میشود و به دلیل اتمام هزینه امکان انتخاب گره دیگری وجود ندارد. (گره‌های انتخابی در این روش E, J)

الگوریتم حریصانه gain/cost:

در این روش برای هر نقطه gain/cost آن نقطه محاسبه شده و گره‌ای با max مقدار انتخاب شده و هزینه آن از بودجه کم شده و این کار ادامه پیدا میکند تا با توجه به بودجه امکان انتخاب گره‌ی دیگری نباشد:

node	gain	cost	gain/cost
A	۱	۳۰۰	۱/۳۰۰
B	۱	٤۰۰	۱/٤۰۰
C	۵	۵۰۰	۵/۵۰۰
D	۲	۵۰۰	۲/۵۰۰
E	۱	۲۰۰	۱/۲۰۰
F	۱	۳۰۰	۱/۳۰۰
G	۱	۲۰۰	۱/۲۰۰
H	۲	۵۰۰	۲/۵۰۰
I	۳	۸۰۰	۳/۸۰۰
J	٦	۸۰۰	٦/۸۰۰
K	۱	۷۰۰	۱/۷۰۰
L	۲	۷۰۰	۲/۷۰۰
M	۲	۹۰۰	۲/۹۰۰
N	٤	٤۰۰	٤/٤۰۰

با توجه به جدول بالا gain/cost برای دو نقطه C,N بیشینه مقدار را دارد، میتوان هر یک از این دو گره را انتخاب کرد. با اتخاب گره‌ی C مقدار gain/cost را برای همه‌ی گره ها دوباره محاسبه میکنیم:

node	gain	cost	gain/cost	(gain {node,C} -gain C)/cost node		
A	۱	۳۰۰	۱/۳۰۰	.		
B	۱	٤۰۰	۱/٤۰۰	.		
C	۵	۵۰۰	۵/۵۰۰			
D	۲	۵۰۰	۲/۵۰۰	.		
E	۱	۲۰۰	۱/۲۰۰	.		
F	۱	۳۰۰	۱/۳۰۰	۱/۳۰۰		
G	۱	۲۰۰	۱/۲۰۰	۱/۲۰۰		
H	۲	۵۰۰	۲/۵۰۰	۲/۵۰۰		
I	۳	۸۰۰	۳/۸۰۰	۳/۸۰۰		
J	٦	۸۰۰	٦/۸۰۰	٦/۸۰۰		
K	۱	۷۰۰	۱/۷۰۰	۱/۷۰۰		
L	۲	۷۰۰	۲/۷۰۰	۲/۷۰۰		
M	۲	۹۰۰	۲/۹۰۰	۲/۹۰۰		
N	٤	٤۰۰	٤/٤۰۰	٤/٤۰۰		

با توجه به جدول بالا، بعد از انتخاب گره C گره‌ی N دارای بیشینه مقدار gain/cost است و با انتخاب آن مشکلی با محدودیت بودجه وجود ندارد و امکان انتخاب گره‌ی دیگری نیست (با انتخاب دو گره C , N بودجه استفاده شده ۹۰۰ هزار تومن است و گره دیگری با بودجه ۱۰۰ هزار تومن وجود ندارد).

د)

الگوریتم CELF دوروش (بدون در نظر گرفتن هزینه و انتخاب براساس gain/cost) را محاسبه میکند و هر یک f(S) بزرگتری داشته باشند در خروجی دارد.

الگوریتم CELF بدون در نظر گرفتن هزینه:

در این الگوریتم gain همه‌ی گره ها در مرحله اول محاسبه شده و در لیست مرتب شده به صورت نزولی ذخیره میشود، در مرحله اول گره ابتدایی لیست انتخاب می‌شود. مطابق شکل زیر در ابتدا گره J انتخاب میشود.

node	gain	cost
J	٦	۸۰۰
C	۵	۵۰۰
N	٤	٤۰۰
I	۳	۸۰۰
D	۲	۵۰۰
H	۲	۵۰۰
L	۲	۷۰۰
M	۲	۹۰۰
A	۱	۳۰۰
B	۱	٤۰۰
E	۱	۲۰۰
F	۱	۳۰۰
G	۱	۲۰۰
K	۱	۷۰۰

برای انتخاب گره دوم، گره بالای لیست انتخاب میشود و {f{node,J}-f{J آن محاسبه میشود اگر از گره بعدی بزرگتر بود به عنوان گره بعدی انتخاب شده در نظر گرفته میشود. با توجه به بودجه در نظر گرفته شده، برای انتخاب گره بعدی امکان انتخاب گره E, G وجود دارد و گره E در بالای لیست است و داریم:

f{J,E}-F{J}>F{G}

پس گره E به عنوان گره دوم انتخاب میشود.

الگوریتم CELF براساس gain/cost:

در این الگوریتم gain/cost همه‌ی گره ها در مرحله اول محاسبه شده و در لیست مرتب شده به صورت نزولی ذخیره میشود، در مرحله اول گره ابتدایی لیست انتخاب می‌شود. مطابق شکل زیر در ابتدا گره C انتخاب میشود.

node	gain	cost	gain/cost
C	۵	۵۰۰	۰٫۰۱
N	٤	٤۰۰	۰٫۰۱
J	٦	۸۰۰	۰٫۰۰۷۵
E	۱	۲۰۰	۰٫۰۰۵
G	۱	۲۰۰	۰٫۰۰۵
D	۲	۵۰۰	۰٫۰۰٤
H	۲	۵۰۰	۰٫۰۰٤
I	۳	۸۰۰	۰٫۰۰۳۸
A	۱	۳۰۰	۰٫۰۰۳۳
F	۱	۳۰۰	۰٫۰۰۳۳
L	۲	۷۰۰	۰٫۰۰۲۹
B	۱	٤۰۰	۰٫۰۰۲۵
M	۲	۹۰۰	۰٫۰۰۲۲
K	۱	۷۰۰	۰٫۰۰۱٤

برای انتخاب گره دوم، گره بعدی لیست انتخاب میشود و benefit/cost آن گره محاسبه شده و با عضو بعدی مقایسه میشود:

f{C,N}-f{C}/400 > 0.0075 => 0.01>0.0075

با توجه به اینکه شرایط بالا برقرا است عضو بعدی انتخاب شده، گره N است و امکان انتخاب گره دیگری وجود ندارد و هزینه نهایی ۹۰۰ هزار تومان است.

ترکیب دو روش فوق:

در روش اول (بدون در نظر گرفتن هزینه) دو گره J , E انتخاب شدند که بودجه استفاده شده برای آنها ۱ ملیون تومن و F(S) برابر ۷ میباشد و در روش دوم (محاسبه gain/cost) دو گره C,N انتخاب شدند که ۹۰۰ هزار تومن از بودجه را مصرف کردند و F(S) برابر ۹ میباشد و با max گرفتن از F(S) دو روش قبلی، دو گره C,N انتخاب در روش دوم انتخاب میشود زیرا مقدار F(S) بزرگتری دارد.

ه)

روش CELF عملکرد بهتری دارد زیرا با توجه به ویژگی های گراف یکی از روش‌های بدن هزینه یا براساس benefit/cost عملکرد بهتری دارند و در این الگوریتم هر دو روش استفاده میشوند و در نهایت هر کدام خروجی

بهتری داشتند به عنوان خروجی نهایی گزارش میکند، که در این گرافها انتخاب براساس روش benefit/cost نتیجه بهتری دارد و f(S) آن برابر با ۹ است و در روش حریصانه اگر تنها انتخاب بدون در نظر گرفتن هزینه باشد f(S) برابر با ۷ میباشد و اگز از روش دوم استفاده شود هر دو الگوریتم خروجی یکسانی دارند.
در روش CELF محاسبات کمتری مورد نیاز است و در هر مرحله نیاز به محاسبه gain تمام نقاط با مجموعه ای که قبلا انتخاب شده نیست، در روش CELF از خاصیت submodular استفاده میشود و با روش lazy hill climbing تنها در مرحله ابتدایی gain تمام نقاط محاسبه میشود.

و)

مسئله تأثیر گذاری به دنبال یافتن مجموعه‌ای از گره‌ها در یک شبکه است که با گسترش اطلاعات یا تأثیر خود، بتوانند تأثیر بیشتری را در شبکه داشته باشند. مسئله تشخیص شیوع به دنبال شناسایی الگوهای گسترش یافته از یک پدیده (مثل بیماری، شایعه، و ...) در شبکه است.
ارتباط میان بیشینه‌سازی تاثیر و تشخیص شیوع این است که در مسئله بیشینه سازی تاثیر، معمولاً سعی در انتخاب مجموعه از گره‌ها داریم تا تاثیر ماکزیمم را بر روی یک موضوع یا اطلاعاتی خاص ایجاد کنیم. این اطلاعات می‌تواند شیوع اطلاعات، ویروس‌ها، تبلیغات، یا هر چیز دیگری باشد. از این نظر، مسئله بیشینه سازی تاثیر وابسته به تشخیص شیوع است، زیرا هدف از هر دو مسئله بهبود تاثیر و انتشار در یک شبکه است.بنابراین، این دو مسئله ممکن است در مواردی به عنوان یکجا در نظر گرفته شوند، به ویژه وقتی که اهمیت شیوع یک اطلاعات یا ویروس در یک شبکه اجتماعی یا شبکه ارتباطی مورد توجه باشد و اطلاعات در مورد زمان شروع و روند شیوع نیز مهم باشد.
در بیشینه سازی تاثیر محدودیت انتخاب گره اولیه وجود دارد و در تشخیص شیوع محدودیت بودجه وجود دارد و در هر دو مسئله باید گره هایی انتخاب شوند که بیشترین تاثیر را در شبکه دارند. در گراف تشخیص شیوع عموما انتخاب هر گره میتواند هزینه داشته باشد و باید محدودیت بودجه در انتخاب گره در نظر گرفته شود و یا باید مجموعه‌ای انتخاب شود که در زمان کوتاه تری یک outbreak را کشف کند و در این تعریف منفی زمان کشف outbreak در نظر گرفته می‌شود.
در برخی موارد، ممکن است از روش‌های مشابهی برای حل این دو مسئله استفاده شود، زیرا هر دو به دنبال تحلیل و پیش‌بینی گسترش در یک شبکه مشترک هستند.
در حالتی که در یک شبکه مسئله تأثیر گذاری حل شده باشد و گره‌های تأثیرگذار شناسایی شده باشند، این اطلاعات ممکن است به تشخیص شیوع یک پدیده کمک کنند. مثلاً، گره‌های تأثیر گذار ممکن است به عنوان نقاط

ابتدایی برای تشخیص شیوع یک بیماری در یک جامعه مورد استفاده قرار گیرند و هر دو مسئله ممکن است در حوزه مدل‌سازی و پیش‌بینی رفتارهای شبکه‌ها مورد استفاده قرار گیرند. این دو مسئله با تأکید بر تحلیل شبکه‌ها و انتشار اطلاعات در آنها ارتباط دارند و از نظر روش‌های حل و اطلاعات مورد استفاده در برخی موارد همپوشانی دارند.

بستگی به جزئیات و ویژگی‌های هر مسئله قابلیت تبدیل به یکدیگر دارند. اما در برخی موارد، می‌توان به اشتراکات و ارتباطات بین این دو مسئله اشاره کرد: هر دو مسئله از طبیعت شبکه‌ها و گراف‌ها بهره می‌برند. در مسئله تأثیر گذاری، گره‌های مهم برای گسترش اطلاعات در یک شبکه مشخص می‌شوند. در مسئله تشخیص شیوع نیز، گره‌های مبتلا یا منطقه‌هایی که به سرعت در آنها پدیده‌ای گسترش می‌یابد، اهمیت دارند.

در هر دو مسئله، شناسایی گره‌های مهم برای انتشار اطلاعات یا تشخیص یک پدیده مشترک است. در مسئله تأثیر گذاری، گره‌هایی که بیشترین تأثیر را دارند، اهمیت دارند. در مسئله تشخیص شیوع، مناطق یا گره‌هایی که به سرعت پدیده گسترش یافته است، اهمیت دارند و این اشتراکات به تبع ماهیت و ویژگی‌های مسائل مختلف ممکن است متفاوت باشند و بستگی به ساختار شبکه و ویژگی‌های مسئله دارد.

سوال چهارم: بیشینه‌سازی تاثیر

الف)

تعداد افراد و تعداد دوستی‌ها به صورت زیر می‌باشد:

Number of nodes: 4039

Number of edges: 88234

ب)

Realization به معنای ایجاد یک گراف واقعی از یک گراف احتمالی است. با احتمالی که هر یال دارد تصمیم می‌گیریم یک یال بماند یا از بین برود، بنابراین گراف باقیمانده یک گراف قطعی است و گراف احتمالی به گراف قطعی تبدیل میشود. ایجاد گراف Realization تنها یکبار انجام نمیشود بلکه تعداد زیادی تکرار می‌کنیم برای اینکه خطا کاهش پیدا کند.

به طور کلی، استفاده از realization یک ویژگی مهم در مطالعه و تحلیل گراف‌ها و تبدیل گراف احتمالاتی به قطعی است و اجازه می‌دهد تا گراف‌ها به عنوان ابزاری برای درک بهتر ساختارهای مختلف در دنیای واقعی استفاده شوند و بسیاری از الگوریتم ها بر مبنای گراف قطعی هستند و استفاده از آنها را راحت می‌سازد.

در تابع create_and_save_realizations_with_activation با دریافت گراف احتمالاتی به تعداد ورودی از آن realization میسازد (با توجه به اطلاعات مسئله احتمال هر یال ۰.۱ در نظر گرفته شده) و در دایرکتوری مدنظر ذخیره میکند و تابع read_realizations گراف‌های realization ساخته شده را از دایرکتوری مدنظر می‌خواند.

ج)

در تابع marginal_gain با دریافت گراف ورودی و مجموعه نودهای انتخاب شده f(S) را محاسبه میکند (تعداد گره‌های قابل دسترسی در گراف با توجه به مجموعه گره‌های انتخاب شده) و برای به دست آوردن f(S) تابع برای هر ۱۰ realization اجرا شده و سپس میانگین گرفته می شود.

انتخاب ۸ گره با بیشترین درجه:

در تابع top_degree_nodes با دریافت گراف ورودی، n گره با بیشترین درجه به همراه تعداد درجه هر یال در خروجی دارد. این تابع برای هر گراف realization اجرا شده است و در نهایت برای گره‌های خروجی میانگین درجه برای هر گره میان همه realization ها محاسبه شده و ۸ گره با بیشترین میانگین درجه در نظر گرفته شده است. در زیر هر یال و میانگین درجه آن در میان realization ها نشان داده شده است.

[('107', 106.8), ('1684', 82.3), ('1912', 76.9), ('3437', 56.1), ('0', 35.5), ('2604', 34.0), ('1730', 34.0), ('2586', 33.0)]

با انتخاب ۸ گره به فوق f(S) به صورت زیر می‌باشد:

Selected Nodes: ('107', '1684', '1912', '3437', '0', '2604', '1730', '2586')

Elapsed time: 2.6521644592285156

f(S): 2988.5

انتخاب ۸ گره تصادفی:

در تابع get_random_nodes بادریافت گراف ورودی، n گره به صورت تصادفی انتخاب می‌شود. این تابع برای همه‌ی realization ها اجرا شده و درنهایت از مجموعه ۸۰ گره که به صورت رندوم انتخاب شده‌اند، ۸ گره انتخاب می‌شود (میتوان با توجه به اینکه در همه realization ها گره‌ها ثابت است و تنها یالها متفاوت است از یکی از realization ها استفاده کرد و ۸ گره به صورت رندوم انتخاب کرد ولی من میخواستم همه realization ها برای انتخاب گره رندوم استفاده بشن که اصلا نیازی ام نیست 🙂)

با انتخاب ۸ گره به صورت تصادفی f(S) به صورت زیر می‌باشد:

Selected Nodes: ['2202', '3347', '1580', '365', '3891', '1905', '187', '3807']

Elapsed time: 1.667137622833252

f(S): 2975.3

د)

الگوریتم حریصانه:

الگوریتم حریصانه، گره ای را اضافه می کند که در حال حاضر بهترین gain را ارائه می دهد که به صورت زیر است:

- با یک مجموعه خالی شروع می کنیم.
- برای تمام گره هایی که در مجموع نیستند، gain آنها را حساب میکنیم و گرهی را با بیشترین gain پیدا می کنیم و آن را به دانه اضافه می کنیم.
- مرحله 2 را تکرار می کنیم تا تعداد مدنظر گره انتخاب شوند.

در این الگوریتم میانگین gain مجموعه انتخاب شده در realization ها در نظر گرفته شده است. نتایج اجرا به صورت زیر است:

Selected Nodes: ['2758', '343', '776', '3980', '4030', '3549', '3382', '3918']

Elapsed time: 21425.70372915268

f(S): 3071.4

الگوریتم CELF:

پیاده‌سازی این الگوریتم به دو جزء تقسیم می‌شود. مؤلفه اول، مانند الگوریتم Greedy، برای هر گره تکرار می‌شود و در نهایت گره ای را که بیشترین gain را دارد، انتخاب می شود. ولی این الگوریتم برای هر گره gain را برای استفاده در جزء دوم نیز ذخیره می کند.

مولفه دوم برای یافتن k-1 باقیمانده تکرار می شود. در هر تکرار، الگوریتم gain گره بالایی لیست را ارزیابی می کند. اگر gain آن با اضافه شدن با مجموعه قبلی گره‌های انتخاب شده در جای خود باقی بماند، آن گره به عنوان گره بعدی انتخاب می شود. در غیر این صورت، گره بالایی جدید ارزیابی می شود.

Selected Nodes: ['2058', '343', '776', '3980', '4030', '3549', '3382', '3579']

Elapsed time: 4340.257801055908

f(S): 3071.4

ه)

select _Top_degree_nodes: 2988.5

Select_random_nodes: 2975.3

Greedy_algorithm: 3071.4

CELF algorithm: 3071.4

دو الگوریتم CELF , حریصانه خروجی مشابهی دارند (در هر مرحله هر دو الگوریتم گره‌ای را انتخاب می‌کنند که بیشترین gain را به مجموعه قبلی اضافه کند)و بالاترین میزان f(S) را دارند (بهترین مقدار ممکن برای F(S) با تعداد گره انتخابی را ارائه میدهند) و روش انتخاب گره به صورت رندوم به گره‌های انتخاب شده بستگی دارد و میتواند خروجی بهتر و یا بدتری نسبت به انتخاب گره‌هایی با بالاترین درجه داشته باشد

الگوریتم CELF از خاصیت submodular استفاده میکند و مقدار gain برای هر گره را در هر مرحله محاسبه نمیکند و بنابراین سریعتر از الگوریتم حریصانه عمل میکند و در هر مرحله بهترین گره که بیشترین مقدار gain را ارائه میدهد، انتخاب می‌کند و در نهایت گره‌های انتخاب شده و f(S) برای هر دو الگوریتم برابر است و این دو الگوریتم خروجی یکسانی دارند.

و)

select _Top_degree_nodes:	gain: 2988.5	time: 2.65
Select_random_nodes:	gain: 2975.3	time: 1.66
Greedy_algorithm:	gain: 3071.4	time: 21425.70
CELF algorithm:	gain: 3071.4	time: 4340.25

الگوریتم حریصانه و CELF بهترین خروجی را دارند ولی الگوریتم CELF سریعتر است، زیرا از رویکرد lazy evaluation برای محاسبه تاثیر گره‌ها استفاده میکند. این به این معنی است که الگوریتم به طور معمول از پیش محاسباتی انجام نمی‌دهد و تنها محاسبات مربوط به تاثیر گره‌ها زمانی انجام می‌شود که این گره‌ها به مجموعه انتخابی اضافه می‌شوند و در این روش gain تمام گره ها ذخیره میشود و با استفاده از ذخیره gain ها و استفاده از خاصیت submodular باعث سرعت این الگوریتم می‌شود.

بعد از الگوریتم CELF و حریصانه، انتخاب گره با بیشترین درجه بهترین خروجی را دارد با اینکه این روش کندتر است و باید درجه همه ی گره ها را محاسبه کند ولی خروجی آن قابل اطمینان تر است و انتخاب گره به صورت تصادفی بسته به گره‌های انتخابی میتواند عملکرد بهتر یا بدتری نسبت به انتخاب براساس درجه داشته باشد.

زمان اجرا به ترتیب برای گره تصادفی، انتخاب گره براساس درجه،CELF و الگوریتم حریصانه است.