Amazon and Twitch Network Analyses

CS 5990: Computational Social Systems
Group Assignment #1

Jeremy Anunwah, Yurii Lebid, Bill Kim, Aumkaareshwar

Initial Analysis: Amazon

- Metrics were stated in documentation and verified via NetworkX
  - Components: one, the entire network is connected
  - Nodes: 334863
  - Edges: 925872
  - Clustering Coefficient: 0.3967
- Additional information, not checked in NetworkX
  - Diameter: 44

Initial Analysis: Twitch

- Metrics were again stated in documentation and verified via NetworkX
  - Components: one, the entire network is connected
  - Size: 168114
  - Edges: 6797557
  - Clustering Coefficient: Not yet known
- No additional information

Methodology

- Size: count the number of nodes in the graph
- Average degree: fast method is to double the edge count and divide by network size
  - Note: each edge contributes 2 to network's total degree
- Average path length: **approximate** via NetworkX
- Clustering coefficient: compute via NetworkX

Methodology

- Calculating average path length is computationally expensive
- As of present, HPC not complete after ~19 hours for each original network
- New method, given *n* nodes:
  - Sample *n* nodes as sources
  - Sample *n* nodes as targets
  - Find average path length of s_1 -> t_1, s_2 -> t_2, ..., etc.
  - Results in O(*n*) paths instead of O(n^2)
- Done for both real and simulated networks

Simulations

- Matching metrics of original networks was guess and check
- Certain trends discovered that sped up experiments
  - Watts-Strogatz: given a fixed parameter β and degree d, an increase in network size caused a consistent increase in average path length
    - clustering coefficient remained relatively constant
  - Barabasi-Albert: given a fixed parameter m, an increase in network size caused a consistent decrease in clustering coefficient
    - average path length decreased slightly

Watts-Strogatz Pseudocode

```
network = NetworkX.Graph()
adjacencyList = [] for n nodes

for each node i
      adjacencyList[i] = [i + 1, ..., i + degree / 2]
end for

for each node i
      for each neighbor j of i
            if randomValue(0.0, 1.0) <= beta
                  loop infinitely
                        k = random node
                        exit loop if i != k, and i and k are not neighbors
                  end loop
            end if
      end for
end for
```

Barabasi-Albert Pseudocode

```
network = NetworkX.Graph()
degrees = [0, 0, ..., 0] for n nodes

for each node i, up to n_0
      for each node j > i, up to n_0
            add edge (i, j) to network
      end for

      degrees[i] = n_0 - 1
end for

for each node i > n_0, up to n
      selectedNodes = random.choices(population = [0, ..., i], weights = degrees[:i], count = m)

      for each selectedNode in selectedNodes:
            add edge (i, selectedNode) to network
            degrees[selectedNode] += 1
      end for

      degrees[i] = m
end for
```

Implementation Validation

- Each algorithm’s implementation was compared against NetworkX implementation
- Analyzed small test networks with varying parameters
- Size, average path length, and clustering coefficient remained equal in all cases

Computed Metrics

	Original Network				Watts-Strogatz		Barabasi-Albert	
Network	Size	Average Degree	Average Path Length	Clustering Coefficient	Average Path Length	Clustering Coefficient	Average Path Length	Clustering Coefficient
Amazon	334863	5.5299	12.9569	0.3967	13.3997	0.4432	10.438	0.0000
Twitch	168114	80.8684	3.8759	0.1599	4.5028	0.1602	3.9999	0.0951

Simulated Network Parameters

Network	Parameter	Value
Amazon (Watts-Strogatz)	β	0.10
	d	6
Amazon (Barabasi-Albert)	m	1
	n_0	20
Twitch (Watts-Strogatz)	β	0.40
	d	60
Twitch (Barabasi-Albert)	m	8
	n_0	500

Results

- Both Watts-Strogatz simulated networks closely matched real networks
- Both Barabasi-Albert simulated networks closely approximated only average path length
  - Clustering coefficient further off due to nature of algorithm
    - Amazon: real coefficient of 0.3967 is high to match
    - Twitch: closer, but n_0 = 500 may be unrealistic

Tasks

- Jeremy Anunwah: presentation, algorithm implementations, simulation experiments
- Yurii Lebid: Verification of metrics for clustering coefficients and average path lengths in parallel computations
- Bill Kim: HPC setup and optimized algorithm design
- Aumkaareshwar: presentation, Contributed to research and analysis.