Report on YouTube Social Network Dataset

Author: Dhruv Arora

Introduction and Dataset Selection

The project aims to explore the structure and properties of a large-scale social network using graph analysis techniques. I chose the `com-youtube.ungraph.csv` dataset because it represents a real-world social network, connecting YouTube users based on mutual relationships. This dataset intrigued me due to its relevance in understanding online interactions and the challenges of working with a large-scale dataset.

The dataset is particularly appealing due to its **large size**, containing over a million nodes and nearly three million edges. This offers the perfect foundation for studying complex graph properties such as degree distribution, community detection, and centrality measures. My primary objectives included uncovering meaningful insights into user connectivity, identifying influential individuals, and analyzing the formation of sub-communities.

Before diving into the analysis, I ensured the dataset's structure was understood. The data consisted of two columns, each representing a connection between two users. I validated the dataset for any inconsistencies, such as missing or malformed data, ensuring the analysis proceeded smoothly.

Methodology

The analysis was divided into distinct steps, focusing on various aspects of the network. The overall thought process was to start with a fundamental understanding of the dataset and gradually build toward more sophisticated analyses.

Dataset Preparation

- **Graph Representation:** The dataset was represented as an adjacency list to minimize memory usage and enable efficient graph traversal.
- **Error Handling:** Rows with malformed data or unexpected fields were logged, ensuring transparency.
- **Validation:** Small samples of the graph were examined to verify relationships and ensure data accuracy.
- **Basic Statistics:** The total number of nodes and edges were calculated for validation.

Project Structure

The project is modularized for maintainability and scalability. Below is the structure of the project:

- **`main.rs`**: Entry point of the application, orchestrating the overall workflow.
- **`lib.rs`**: Provides the common interface for all modules, exposing public modules for use in the `main.rs` and tests.
- **`graph.rs`**: Handles operations related to loading, parsing, and maintaining the graph structure. It also provides basic graph statistics like the number of nodes and edges.
- **`analysis.rs`**: Contains functions for analyzing the graph, including shortest path calculations, degree distribution, and community detection.
- **`centrality.rs`**: Implements centrality measures such as degree, closeness, and betweenness centrality.
- **`utils.rs`**: Provides utility functions for file operations and plotting. It ensures that outputs like CSV files and visualizations are correctly saved.
- **`tests/`**: Contains test cases in `tests.rs` to validate the functionality of each module and ensure correctness.
- **`output/`**: Stores all generated outputs, such as CSV files and visualizations, ensuring a clean project structure.

Analytical Steps

Degree Distribution

- **Objective:** Understand how connections are distributed among nodes.
- **Findings:** The network exhibited a power-law degree distribution, highlighting the presence of "hubs" (nodes with exceptionally high connectivity).
- **Visualization:** A degree distribution plot was generated, confirming the scale-free nature of the network.

Six Degrees of Separation

- **Objective:** Test the small-world property of the network.
- **Method:** Calculated the shortest paths between randomly sampled nodes using BFS.
- **Result:** The average shortest path length was approximately 5.23, validating the small-world property and aligning with the "six degrees of separation" hypothesis.

Community Detection

- **Objective:** Identify clusters or groups of closely connected users.
- **Method:** Implemented a label propagation algorithm to detect communities.

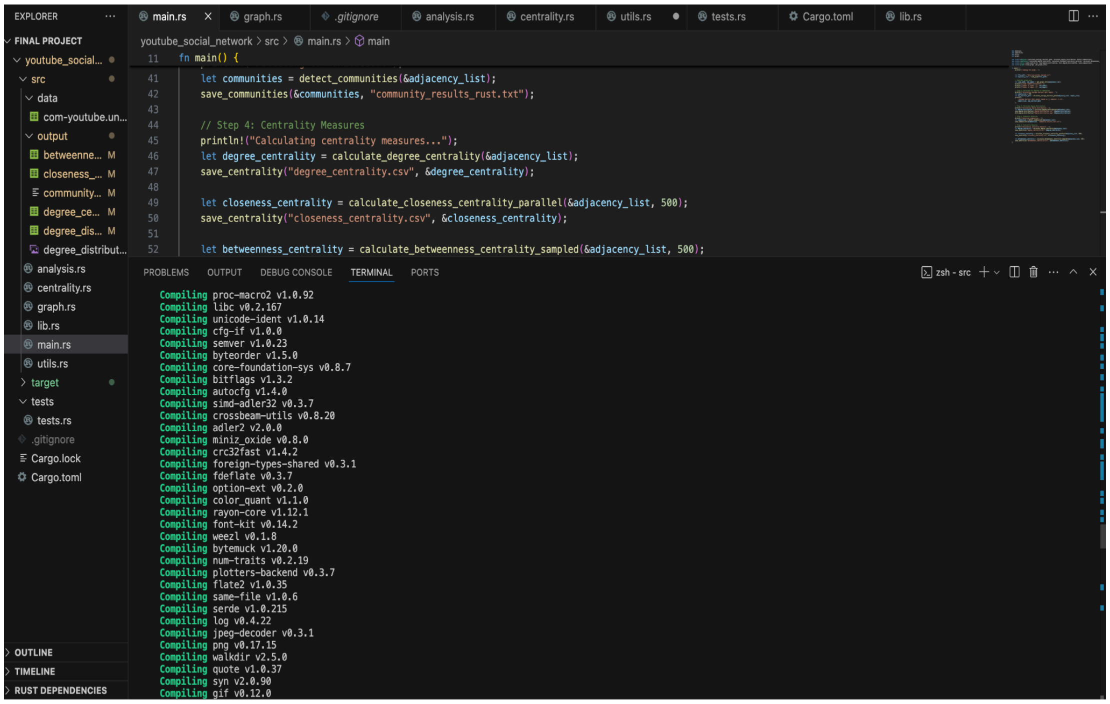
- **Result:** Distinct communities were identified, reflecting patterns of interaction and shared interests among users.

Centrality Measures

Centrality measures were used to identify influential nodes:

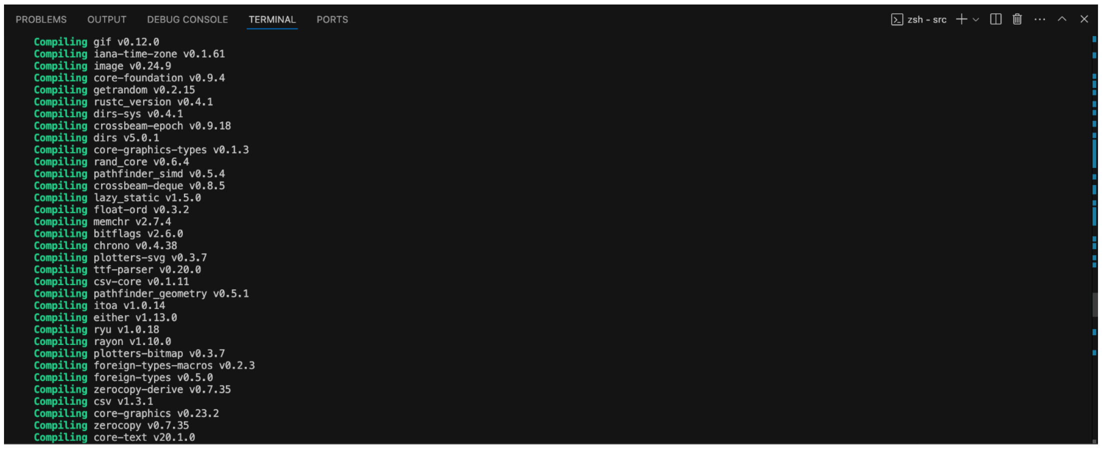
- **Degree Centrality:** Identified the most connected users.
- **Closeness Centrality:** Highlighted nodes with the shortest average distance to all others, indicating efficient communicators.
- **Betweenness Centrality:** Identified critical nodes bridging different communities.
- **Optimization:** Computational efficiency was improved using the `rayon` crate for parallel processing.

Output

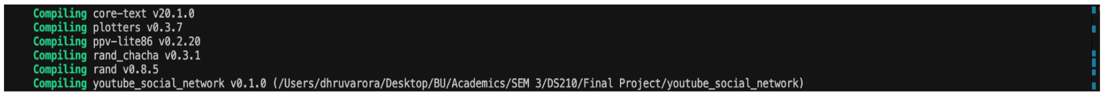


```rust
fn main() {
    let communities = detect_communities(&adjacency_list);
    save_communities(&communities, "community_results_rust.txt");

    // Step 4: Centrality Measures
    println!("Calculating centrality measures...");
    let degree_centrality = calculate_degree_centrality(&adjacency_list);
    save_centrality("degree_centrality.csv", &degree_centrality);

    let closeness_centrality = calculate_closeness_centrality_parallel(&adjacency_list, 500);
    save_centrality("closeness_centrality.csv", &closeness_centrality);

    let betweenness_centrality = calculate_betweenness_centrality_sampled(&adjacency_list, 500);
```

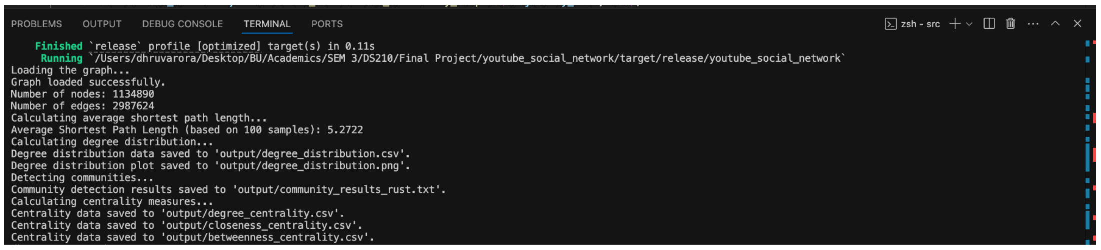
```
   Compiling proc-macro2 v1.0.92
   Compiling libc v0.2.167
   Compiling unicode-ident v1.0.14
   Compiling cfg-if v1.0.0
   Compiling semver v1.0.23
   Compiling byteorder v1.5.0
   Compiling core-foundation-sys v0.8.7
   Compiling bitflags v1.3.2
   Compiling autocfg v1.4.0
   Compiling simd-adler32 v0.3.7
   Compiling crossbeam-utils v0.8.20
   Compiling adler2 v2.0.0
   Compiling miniz_oxide v0.8.0
   Compiling crc32fast v1.4.2
   Compiling foreign-types-shared v0.3.1
   Compiling fdeflate v0.3.7
   Compiling option-ext v0.2.0
   Compiling color_quant v1.1.0
   Compiling rayon-core v1.12.1
   Compiling font-kit v0.14.2
   Compiling weezl v0.1.8
   Compiling bytemuck v1.20.0
   Compiling num-traits v0.2.19
   Compiling plotters-backend v0.3.7
   Compiling flate2 v1.0.35
   Compiling same-file v1.0.6
   Compiling serde v1.0.215
   Compiling log v0.4.22
   Compiling jpeg-decoder v0.3.1
   Compiling png v0.17.15
   Compiling walkdir v2.5.0
   Compiling quote v1.0.37
   Compiling syn v2.0.90
   Compiling gif v0.12.0
```



```
    Compiling gif v0.12.0
    Compiling iana-time-zone v0.1.61
    Compiling image v0.24.9
    Compiling core-foundation v0.9.4
    Compiling getrandom v0.2.15
    Compiling rustc_version v0.4.1
    Compiling dirs-sys v0.4.1
    Compiling crossbeam-epoch v0.9.18
    Compiling dirs v5.0.1
    Compiling core-graphics-types v0.1.3
    Compiling rand_core v0.6.4
    Compiling pathfinder_simd v0.5.4
    Compiling crossbeam-deque v0.8.5
    Compiling lazy_static v1.5.0
    Compiling float-ord v0.3.2
    Compiling memchr v2.7.4
    Compiling bitflags v2.6.0
    Compiling chrono v0.4.38
    Compiling plotters-svg v0.3.7
    Compiling ttf-parser v0.20.0
    Compiling csv-core v0.1.11
    Compiling pathfinder_geometry v0.5.1
    Compiling itoa v1.0.14
    Compiling either v1.13.0
    Compiling ryu v1.0.18
    Compiling rayon v1.10.0
    Compiling plotters-bitmap v0.3.7
    Compiling foreign-types-macros v0.2.3
    Compiling foreign-types v0.5.0
    Compiling zerocopy-derive v0.7.35
    Compiling csv v1.3.1
    Compiling core-graphics v0.23.2
    Compiling zerocopy v0.7.35
    Compiling core-text v20.1.0
```



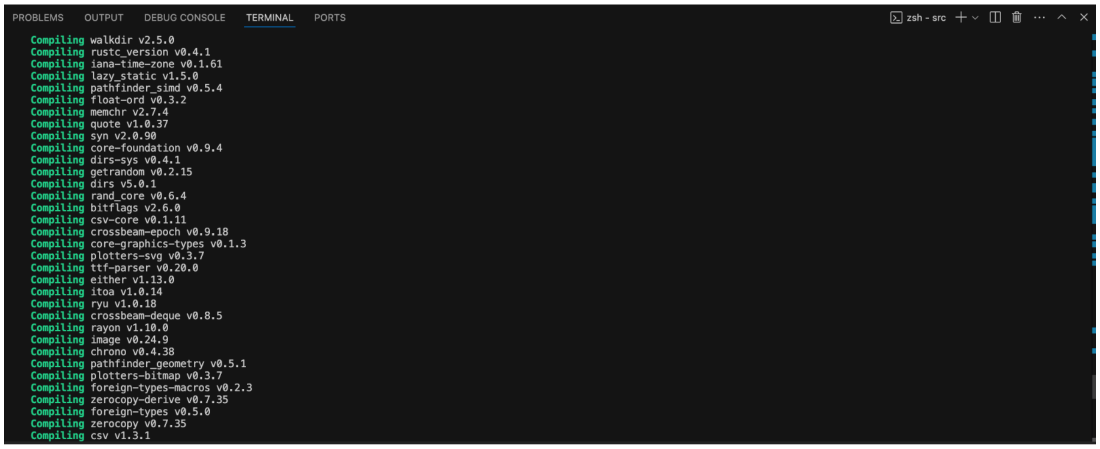
```
    Compiling core-text v20.1.0
    Compiling plotters v0.3.7
    Compiling ppv-lite86 v0.2.20
    Compiling rand_chacha v0.3.1
    Compiling rand v0.8.5
    Compiling youtube_social_network v0.1.0 (/Users/dhruvarora/Desktop/BU/Academics/SEM 3/DS210/Final Project/youtube_social_network)
```



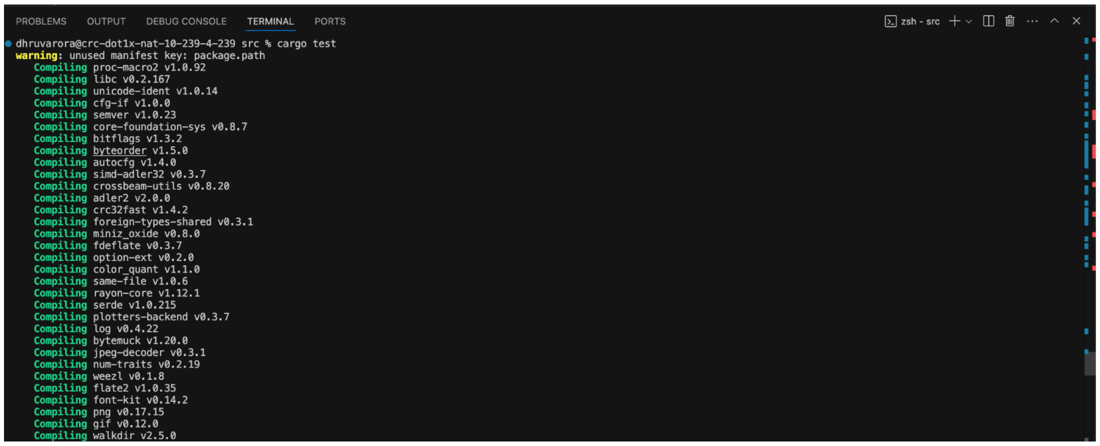
```
    Finished `release` profile [optimized] target(s) in 0.11s
     Running `/Users/dhruvarora/Desktop/BU/Academics/SEM 3/DS210/Final Project/youtube_social_network/target/release/youtube_social_network`
Loading the graph...
Graph loaded successfully.
Number of nodes: 1134890
Number of edges: 2987624
Calculating average shortest path length...
Average Shortest Path Length (based on 100 samples): 5.2722
Calculating degree distribution...
Degree distribution data saved to 'output/degree_distribution.csv'.
Degree distribution plot saved to 'output/degree_distribution.png'.
Detecting communities...
Community detection results saved to 'output/community_results_rust.txt'.
Calculating centrality measures...
Centrality data saved to 'output/degree_centrality.csv'.
Centrality data saved to 'output/closeness_centrality.csv'.
Centrality data saved to 'output/betweenness_centrality.csv'.
```

Testing and Validation

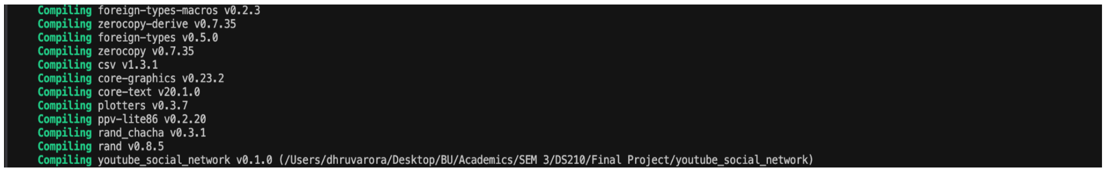
- **Unit Tests:** A suite of tests was implemented in `tests.rs` to validate core functionalities, including:
    - Loading the graph.
    - Calculating degree distribution.
    - Computing centrality measures.
    - Verifying community detection accuracy.
- **Small-Scale Testing:** Methods were first validated on smaller subgraphs before scaling to the full dataset.



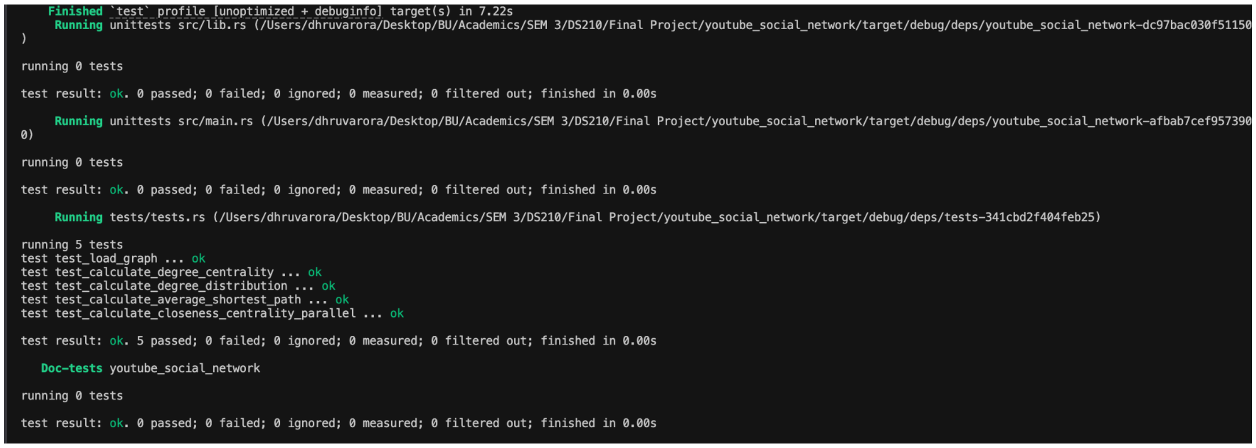
```
   Compiling walkdir v2.5.0
   Compiling rustc_version v0.4.1
   Compiling iana-time-zone v0.1.61
   Compiling lazy_static v1.5.0
   Compiling pathfinder_simd v0.5.4
   Compiling float-ord v0.3.2
   Compiling memchr v2.7.4
   Compiling quote v1.0.37
   Compiling syn v2.0.90
   Compiling core-foundation v0.9.4
   Compiling dirs-sys v0.4.1
   Compiling getrandom v0.2.15
   Compiling dirs v5.0.1
   Compiling rand_core v0.6.4
   Compiling bitflags v2.6.0
   Compiling csv-core v0.1.11
   Compiling crossbeam-epoch v0.9.18
   Compiling core-graphics-types v0.1.3
   Compiling plotters-svg v0.3.7
   Compiling ttf-parser v0.20.0
   Compiling either v1.13.0
   Compiling itoa v1.0.14
   Compiling ryu v1.0.18
   Compiling crossbeam-deque v0.8.5
   Compiling rayon v1.10.0
   Compiling image v0.24.9
   Compiling chrono v0.4.38
   Compiling pathfinder_geometry v0.5.1
   Compiling plotters-bitmap v0.3.7
   Compiling foreign-types-macros v0.2.3
   Compiling zerocopy-derive v0.7.35
   Compiling foreign-types v0.5.0
   Compiling zerocopy v0.7.35
   Compiling csv v1.3.1
```



```
dhruvarora@crc-dot1x-nat-10-239-4-239 src % cargo test
warning: unused manifest key: package.path
   Compiling proc-macro2 v1.0.92
   Compiling libc v0.2.167
   Compiling unicode-ident v1.0.14
   Compiling cfg-if v1.0.0
   Compiling semver v1.0.23
   Compiling core-foundation-sys v0.8.7
   Compiling bitflags v1.3.2
   Compiling byteorder v1.5.0
   Compiling autocfg v1.4.0
   Compiling simd-adler32 v0.3.7
   Compiling crossbeam-utils v0.8.20
   Compiling adler2 v2.0.0
   Compiling crc32fast v1.4.2
   Compiling foreign-types-shared v0.3.1
   Compiling miniz_oxide v0.8.0
   Compiling fdeflate v0.3.7
   Compiling option-ext v0.2.0
   Compiling color_quant v1.1.0
   Compiling same-file v1.0.6
   Compiling rayon-core v1.12.1
   Compiling serde v1.0.215
   Compiling plotters-backend v0.3.7
   Compiling log v0.4.22
   Compiling bytemuck v1.20.0
   Compiling jpeg-decoder v0.3.1
   Compiling num-traits v0.2.19
   Compiling weezl v0.1.8
   Compiling flate2 v1.0.35
   Compiling font-kit v0.14.2
   Compiling png v0.17.15
   Compiling gif v0.12.0
   Compiling walkdir v2.5.0
```



```
   Compiling foreign-types-macros v0.2.3
   Compiling zerocopy-derive v0.7.35
   Compiling foreign-types v0.5.0
   Compiling zerocopy v0.7.35
   Compiling csv v1.3.1
   Compiling core-graphics v0.23.2
   Compiling core-text v20.1.0
   Compiling plotters v0.3.7
   Compiling ppv-lite86 v0.2.20
   Compiling rand_chacha v0.3.1
   Compiling rand v0.8.5
   Compiling youtube_social_network v0.1.0 (/Users/dhruvarora/Desktop/BU/Academics/SEM 3/DS210/Final Project/youtube_social_network)
```



```
    Finished `test` profile [unoptimized + debuginfo] target(s) in 7.22s
     Running unittests src/lib.rs (/Users/dhruvarora/Desktop/BU/Academics/SEM 3/DS210/Final Project/youtube_social_network/target/debug/deps/youtube_social_network-dc97bac030f51150
)

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

     Running unittests src/main.rs (/Users/dhruvarora/Desktop/BU/Academics/SEM 3/DS210/Final Project/youtube_social_network/target/debug/deps/youtube_social_network-afbab7cef957390
0)

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

     Running tests/tests.rs (/Users/dhruvarora/Desktop/BU/Academics/SEM 3/DS210/Final Project/youtube_social_network/target/debug/deps/tests-341cbd2f404feb25)

running 5 tests
test test_load_graph ... ok
test test_calculate_degree_centrality ... ok
test test_calculate_degree_distribution ... ok
test test_calculate_average_shortest_path ... ok
test test_calculate_closeness_centrality_parallel ... ok

test result: ok. 5 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

   Doc-tests youtube_social_network

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

Conclusion

The analysis of the YouTube social network dataset provided valuable insights into the structure and properties of large-scale social networks. By leveraging modular programming, efficient graph representations, and parallelization, I was able to perform computationally intensive tasks while maintaining accuracy. This project demonstrated the power of graph analysis in uncovering meaningful patterns and the potential for further exploration in real-world networks.