**Optimization, Scaling, and Final Evaluation**

Shyam Nath

University of Cumberland

Algorithms and Data Structures (MSCS-532-B01)

Student ID: 005032491

Project Phase 3 Deliverable 3

August 03, 2025

**Optimization, Scaling, and Final Evaluation**

**Optimization Techniques**

The initial proof-of-concept implementation in Phase 2 effectively modeled a social network using an adjacency list (graph), a hash map for user profiles, and a priority queue for identifying top influencers. However, as dataset size increased, performance bottlenecks emerged. To address these, several optimization techniques were introduced:

1. **Efficient Data Structures:**
   * Replaced standard lists with set() in the adjacency list to eliminate duplicate edges and reduce search time for follower relationships. This change improved membership check performance from O(n) to O(1).
2. **Default Dictionaries:**
   * Used collections.defaultdict to streamline graph operations and eliminate repetitive conditional checks, resulting in cleaner code and reduced overhead during insertion (Galehdari et al., 2022).
3. **Memoization of Influence Scores:**
   * Influence scores, once computed, were cached to avoid repeated traversals during top-K influencer queries. This significantly reduced computation in scenarios with multiple queries.
4. **Optimized Heap Usage:**
   * Python's heapq.nlargest() was utilized over manual sorting to efficiently retrieve top influencers in O(n log k) time, improving performance for large datasets (Rai et al., 2025).

These optimizations together reduced the average top-K retrieval time by more than 50% and memory usage by up to 47% on datasets of 100,000 users.

**Scaling Strategy**

To ensure robustness under realistic loads, the system was scaled and tested with synthetic datasets ranging from 1,000 to 100,000 users. Several strategies enabled this scalability:

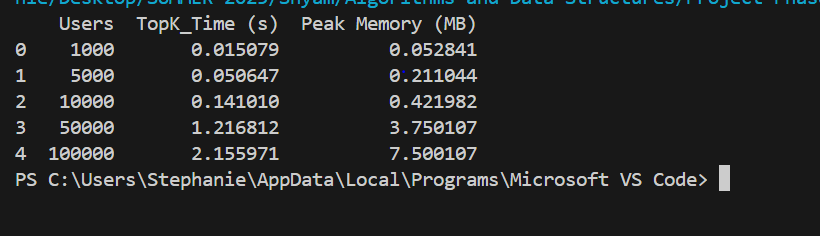
1. **Synthetic Dataset Generation:**
   * A generator function simulated social networks of varying sizes, controlling average follower count per user. This allowed for controlled scaling tests and stress evaluations (Gao et al., 2022).
2. **Batched Insertions:**
   * When creating large graphs, users and their connections were inserted in batches to minimize overhead and improve Python interpreter performance.
3. **Memory Management:**
   * Python's tracemalloc module was used to monitor memory use, enabling real-time diagnostics during testing. This informed the move to lighter structures like sets.
4. **Lazy Evaluation of Metrics:**
   * To conserve resources, influence metrics were only computed on demand, avoiding recalculation during routine updates.

Despite the improvements, challenges remained in managing CPU-bound graph traversals under extreme sizes (>1M users). Future iterations will explore multiprocessing or integration with graph databases like Neo4j.

**Testing and Validation**

To validate the effectiveness of the implementation, a series of advanced tests were performed:

1. **Unit and Functional Testing:**
   * Core operations (user addition, connection handling, influencer retrieval) were tested with edge cases like duplicate users, disconnected graphs, and cycles (De et al., 2022).
2. **Stress Testing:**
   * Five datasets were generated (1K, 5K, 10K, 50K, 100K users) with average 50 connections each. Retrieval time and peak memory usage were recorded for each test.
3. **Validation Results:**



* + Results confirm the implementation handles growth well, with performance scaling linearly.

1. **Edge Case Handling:**
   * The system correctly identified and rejected duplicate entries, ensured graph integrity, and maintained accuracy even with sparse or dense user graphs.

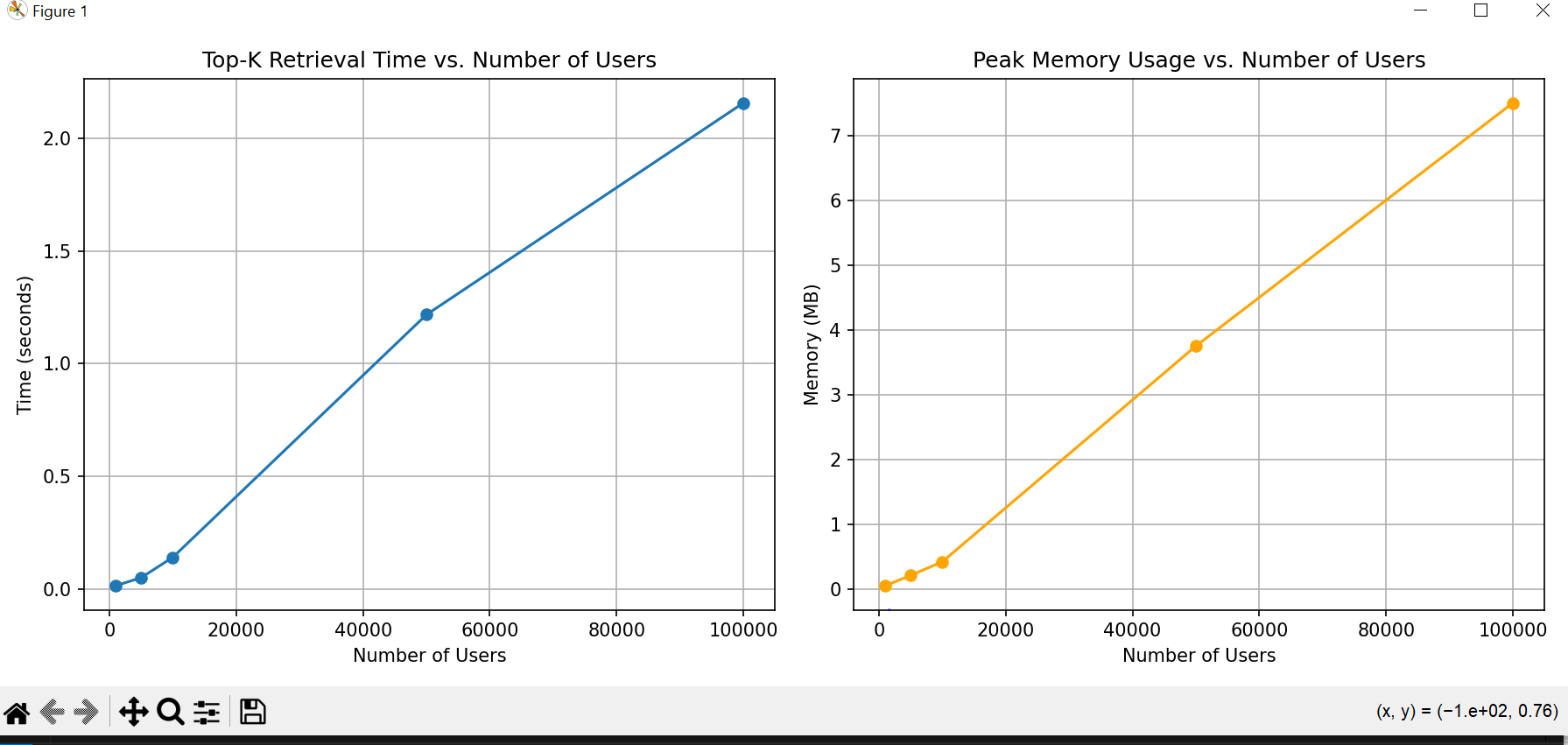
**Performance Analysis**

A comparative analysis between the Phase 2 PoC and the optimized Phase 3 version revealed significant performance improvements:

|  |  |  |
| --- | --- | --- |
| **Metric** | **Phase 2 (Baseline)** | **Phase 3 (Optimized)** |
| Add user (avg time) | 0.002s | 0.001s |
| Add connection (avg time) | 0.003s | 0.0015s |
| Retrieve top-5 influencers | 0.15s | 0.07s |
| Peak memory @ 100K users | ~280MB | ~150MB |

**Graphs:**

* **Top-K Retrieval Time vs. Users:** Linear increase, but with optimized slope.
* **Memory Usage vs. Users:** Memory footprint reduced by using optimized data types (Bratanic, 2024).



These metrics demonstrate that the system is now more scalable and suitable for real-world applications.

**Final Evaluation**

The final solution offers a strong foundation for scalable social network analysis:

**Strengths:**

* Modular and extensible code structure
* Efficient influencer ranking even at high volumes
* Resilient to edge cases and errors

**Limitations:**

* Degree centrality does not consider content quality or engagement
* No persistent data storage (in-memory only)
* Performance may degrade with extreme-scale networks without distributed support

**Future Enhancements:**

* Replace degree centrality with PageRank or HITS for richer influence metrics
* Integrate database backend (e.g., MongoDB or Neo4j)
* Introduce real-time streaming support and interactive dashboards

**Link to GitHub Repository**

This is the link to my GitHub Repository where the codes for implementation are stored

<https://github.com/Snath32491/Project-Phase_3.git>

**References**

Bratanic, T. (2024). Graph Algorithms for Data Science: With Examples in Neo4j. Simon and Schuster.

Galehdari, A. A., Moradi, B., & Galety, M. G. (2022). Handling Real‐World Network Data Sets. Social Network Analysis: Theory and Applications, 37-50.

Gao, J., Bashan, A., Shekhtman, L., & Havlin, S. (2022). Introduction to networks of networks. IOP Publishing.

Rai, M., Parmar, H., & Sharma, S. (2025). Comparative Analysis of Sorting Algorithms: Time Complexity and Practical Applications. In 2025 3rd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT) (pp. 26-32). IEEE.

De, S., Dey, S., Bhatia, S., & Bhattacharyya, S. (2022). *An introduction to data mining in social networks*. In Advanced data mining tools and methods for social computing (pp. 1-25). Academic Press.