Louvain Clustering for Big Data Graph Visual Analytics

David Gauldie, Scott Langevin, Peter Schretlen, David Jonker, Neil Bozowsky, William Wright*

Oculus Info Inc.

ABSTRACT

Challenges with using graphs to visualize extremely large entity-relationship datasets include visibility, usability and high degree nodes. Visual aggregation techniques, tools and easily tailorable components are needed that will support answering analytical questions with data description, characterization and interaction without loss of information. We present two case studies of prototype implementations of JavaScript browser-based visualization tools leveraging the Louvain clustering algorithm. Two "big data" datasets were used to test aggregation of large networks to reveal communities and answer analytical questions.

Keywords: Large Data Visualization, Graph/Network Data, Data Clustering, Distributed Computing.

1 BACKGROUND

Challenges in visualizing large entity-relationship datasets are well known. "Hairballs" resulting from trying to portray even just a fraction of such datasets are difficult and time consuming to explore and understand. Analytical questions of who is connected to whom are difficult to answer. New visual aggregation techniques and easily tailorable components are needed for characterization and interaction without loss of information.

As an alternative to graph (i.e. node-link) visualizations, semantic substrates [1] improve user ability to understand entity attributes and pair-wise relationships. However, this comes at the cost of making it harder to see the graph topology, which is important to be able to identify community structure.

In a network, communities are sub-units of nodes that are highly interconnected. These may be functional groups such as a meme in an information network or an emerging research thread in a co-citation network. An approach to node-link visualization that deals with scale and can preserve community structure is aggregation. Communities can be collapsed and represented as meta-nodes. Optimal partitioning of such datasets into densely connected communities, where relationships are sparse between nodes in different communities, is an intractable problem. Thus algorithms for community detection aim to find a balance between the quality of the partitions and the required compute time.

Louvain clustering [2] provides a simple heuristic method based on modularity optimization to extract hierarchical community structure of large networks. While other algorithms such as Multi-Attribute Clustering (MAC) [3] can provide better computational performance and more control over the resolution of summary and results in aggregate nodes, in our tests Louvain produced higher quality results. Louvain aggregated graphs more clearly showed distinct communities and resulted in less links, making it easier to understand relationships than with MAC.

* email: {dgauldie, slangevin, pschretlen, djonker, nbozowsky, bwright} @oculusinfo.com

2 APPROACH

We present two case studies of prototype implementations of browser-based visual analytics tools leveraging the Louvain clustering algorithm. These are implemented in ApertureJS, a new open source, JavaScript visual analytics library [4]. Two large datasets—CharityNet and Bitcoin—were used to test aggregation of large networks to reveal communities and answer analytical questions. Implementation performance is of interest to support interactive visualizations for time-sensitive, actionable analysis.

2.1 Community Structure and Donation Patterns in CharityNet

CharityNet is a big data graph of anonymized charities and donation transactions recorded over a two year period. It contains 1.8M donors (nodes), 6K charities (nodes) and 3.3M donations (links or edges). Our analytic task was to identify strategies for increasing a charity's level of support by investigating charity and donor community structure and donation patterns. The analytical questions were:

- Characterize who donated to a charity, from where, how much and how often?
- How is the charity performing relative to its peers?
- Who gives money to the charity's peers, but not the charity?

To support the user questions and analytical tasks performed on the dataset, the CharityNet application visualizes a root charity, all the donors who have donated to that charity, and all the other charities to which they have also donated (Figure 1).

Our implementation of Louvain, was single threaded and ran in memory. It was necessary to compute in advance and cache results in a cluster member table.

2.2 Financial Forensics Analysis for Bitcoin

Bitcoin is an anonymous, stateless, encrypted online currency with known ties to black markets, illicit drugs, and illegal gambling. It supports a large anonymous online marketplace. We tested with a dataset of 5.4M source IDs (nodes) and 37.45M transactions (links or edges) to 6.3M destination IDs (nodes).

For transactions of interest, our analytic task was to describe the user community around the source and destination addresses, and summarize the transaction activity of that community around the time of the transaction. This type of analysis might be triggered by theft, unusual transfer, or activity around market swings.

This data proved to be a good fit for "Influent", a browser-based application we have developed using ApertureJS [4]. Influent is specialized for visualizing financial transaction flows. In the case of money laundering, for example, flow that branches to multiple accounts and later flows into the same account, suggesting control by a single entity, is readily observable.

Louvain clustering is especially useful on the Bitcoin dataset where there are few attributes and so limits attribute based clustering. For working with Bitcoin data in Influent, Louvain aggregation provides particular utility through clustering broad search results, or when hitting a high degree node (thousands or millions of links) while tracing and linking through connections. To compute community structures, a distributed processing configuration was implemented in Spark on a high memory

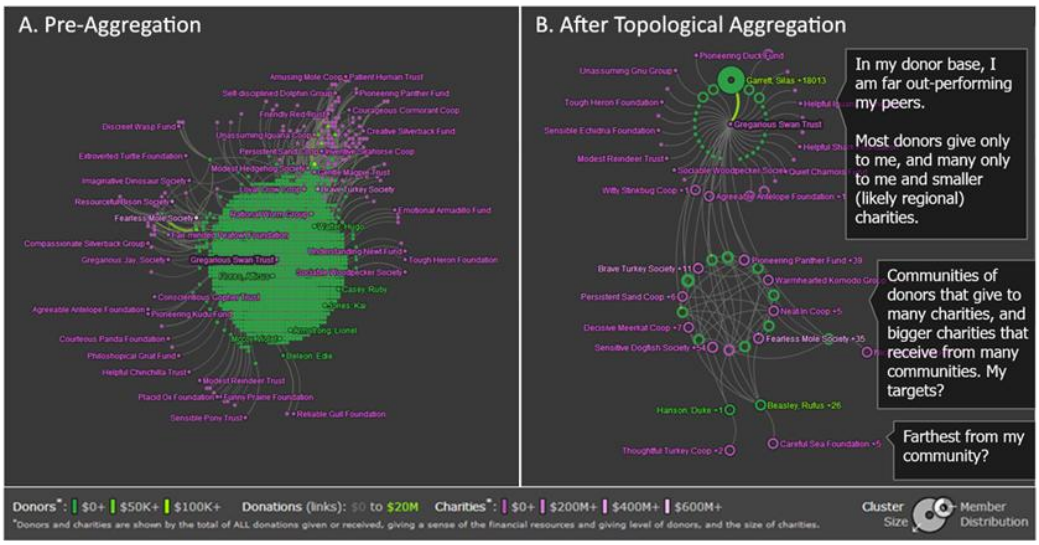

Figure 1: Example CharityNet analysis of donation patterns surrounding "Gregarious Swan Trust". In A, it is very difficult to see how this charity is performing relative to peers. After applying Louvain aggregation to the same community in B, donation patterns emerge.

cluster. Data is first staged in Hadoop Distributed File System (HDFS) or alternatively in Hive tables. Output from the algorithm is stored in HDFS as follows:

1. **Giraph output**. For each clustering iteration, there is an Apache Giraph job that outputs a HDFS file with fields: id, community id, internal weight, list of edges to communities.
2. **Map Reduce output**. For each clustering iteration there is a Map Reduce output HDFS file that matches the required input for the Giraph job, and represents a community compressed version of the graph. Each node represents an entire community.

The processed data is then inserted into Influent dataview tables, currently implemented in MS SQL Server. As a next step, we will be using Cloudera Impala which will provide interactive response times expected to be ~2 seconds.

3 Performance Analysis

3.1 First Implementation (Louvain over CharityNet)

The CharityNet application visualizes a root charity, all the donors who have donated to that charity, and all the other charities to which they have donated. Louvain aggregation was performed on these subsets, ranging in size from just a few nodes and links to ones with approximately 200K nodes and 240K links.

With this single-threaded implementation of Louvain aggregation running on a single four-core processor with 24 GB of RAM, processing time was recorded for each subset. While processing was not completed for the entire dataset, we estimate it would have required approximately 45 hours of continuous processing time for the entire dataset of 1.8M nodes (1.6 GB).

3.2 Second Implementation (Louvain over Bitcoin)

Using a distributed implementation of Louvain running over the Bitcoin dataset required approximately 40 minutes to complete processing of 11.7M nodes (3.6 GB). This was accomplished using an 8 node cluster, with 24 cores and 190 GB of RAM per node. Processing power was underutilized, however, as only 12 threads were used, so faster times are expected in the future.

The Louvain output for the Bitcoin dataset is post-processed with a set of Python scripts to transform the data into the Influent dataview tables, using a single four-core processor with 24GB of RAM. The data is first transformed into a denormalized mapping of entities to all clusters to which the entity belongs, for all hierarchies (for all Louvain iterations done). For the Bitcoin Louvain output this denormalization is approximately 5 minutes.

We then use another script to process the denormalized mappings to a renormalized table that matches the Influent dataview table. This renormalization step also takes approximately 5 minutes to complete. At this point there are over 42 million rows of data, which we are currently manually adding to the Influent dataview table in MS SQL Server, using a flat file import in SQL Server Management Studio. This import, loading the data over a local network, takes approximately one hour.

4 Conclusion

Significant performance improvements have been achieved using high memory cluster configurations for implementation of the Louvain aggregation algorithm enabling timely visual analytics on significantly larger datasets. Work is underway to address automation of pre- and post-processing steps for less labor intensive data staging and reformatting.

Next we will be exploring batch-processing/real-time thresholds to determine when to re-compute by batch or on the fly. The tradeoffs will vary with available computing resources and the volume of data. We are now benchmarking the decision space.

Acknowledgement

Thanks to Sotera Defense for the distributed Louvain implementation. This study was supported by Defense Advanced Research Projects Agency (DARPA) under Contract Number FA8750-12-C-0317. The views, opinions, and findings contained in this report are those of the authors and should not be construed as an official Department of Defense position, policy, or decision.

References

[1] Shneiderman, B. et al. **Network Visualization by Semantic Substrates**, IEEE TVCG, Vol. 12, No. 5, 2006.

[2] Blondel, V., et al, **Fast Unfolding of Communities in Large Networks**, Statistical Mechanics: Theory + Experiment, No 10, 2008

[3] Tian, Y., R., Hankins, and J. Patel, **Efficient Aggregation for Graph Summarization**, SIGMOD, 2008.

[4] Jonker, D., et al, **Aperture: An Open Web 2.0 Visualization Framework**, HICSS, 2013.