A SCALABLE SIGNAL PROCESSING ARCHITECTURE FOR MASSIVE GRAPH ANALYSIS

Benjamin A. Miller[1]*, Nicholas Arcolano*[1]*, Michelle S. Beard*[1]*, Jeremy Kepner*[1]*, Matthew C. Schmidt*[1]*, Nadya T. Bliss*[1] *and Patrick J. Wolfe*[2]

[1]Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA, 02420
{bamiller, arcolano, michelle.beard, kepner, matthew.schmidt, nt}@ll.mit.edu
[2]Statistics and Information Sciences Laboratory, Harvard University, Cambridge, MA, 02138
wolfe@stat.harvard.edu

ABSTRACT

In many applications, it is convenient to represent data as a graph, and often these datasets will be quite large. This paper presents an architecture for analyzing massive graphs, with a focus on signal processing applications such as modeling, filtering, and signal detection. We describe the architecture, which covers the entire processing chain, from data storage to graph construction to graph analysis and subgraph detection. The data are stored in a new format that allows easy extraction of graphs representing any relationship existing in the data. The principal analysis algorithm is the partial eigendecomposition of the modularity matrix, whose running time is discussed. A large document dataset is analyzed, and we present subgraphs that stand out in the principal eigenspace of the time-varying graphs, including behavior we regard as clutter as well as small, tightly-connected clusters that emerge over time.

***Index Terms*—** Graph theory, large data analysis, processing architectures, residuals analysis, emergent behavior

1. INTRODUCTION

As data collection capabilities improve, the amount of data available for analysis rapidly increases. While improved processing enables us to work with much larger datasets, it is at the same time important to develop scalable algorithms and architectures to handle and analyze massive data.

In many applications, the data of interest can be represented as a graph. A graph $G = (V, E)$ is a pair of sets: a set of vertices, V, representing entities, and a set of edges, E, that represent relationships between the entities. This data representation is used in a wide variety of domains, from the social sciences to physics and engineering. While convenient and intuitive, the analysis of graphs is complicated, as they are combinatorial structures of non-Euclidean data and, thus, cannot be exactly analyzed in the context of Euclidean vector spaces. To address this, recent work has focused on developing a statistical detection theory framework for graphs, akin to that for Euclidean data [1]. As more applications use data in the form of graphs, a focus on detectability of anomalies and scalability of architectures and algorithms will become increasingly important.

In this paper, we introduce a signal processing architecture specifically designed for the analysis of massive graphs. This architecture encompasses the entire graph processing procedure, from storage of the raw data to extraction of relational structure to analysis of the resulting graph. Building on recently-developed technologies, this architecture provides a framework with which we can easily analyze large datasets containing complex relationships.

The remainder of this paper is organized as follows. In Section 2, we describe the architecture and discuss the data storage format, the graph construction procedure, and the analysis algorithms and their complexity. Section 3 introduces our dataset of interest—a large document database—and outlines graphs of interest derived from this dataset. In Section 4, we analyze the data using the algorithms described in Section 2, and describe emerging clusters found in the data, as well as some "clutter" structures that can obscure more interesting behavior. In Section 5 we summarize and outline future research.

2. SYSTEM ARCHITECTURE

Our processing chain consists of 3 stages. First, the data are stored in the D4M format [2]. From this data, we construct graphs representing a variety of relationships. We then run analysis algorithms on the resulting graphs. In this section we describe each component in detail.

2.1. Data into D4M

Given a large dataset, D4M provides a convenient, intuitive interface for accessing subsets of the data. In this format, the data are stored in a 2-dimensional associative array, which is a sparse matrix whose rows and columns are indexed by keys rather than integers. Consider, for example, an associative array holding information about papers at a conference. If each record contains the paper title, authors, and session (e.g., My Paper Title, A. Researcher, Session 5), there will be nonzero values in row "title/My Paper Title" only in columns "author/Researcher, A." and "session/Session 5".

To extract a subset of the data, we can index into ranges of the associative array, just as is done with matrices in Matlab. For example, to extract the work by all authors with the name Researcher in the example database, we simply index into the column range "author/Researcher, A,:,author/Researcher, Z,". This operation returns an associative array with its columns restricted to the authors of interest.

This work is supported by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Contract FA8721-05-C-0002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.
Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA or the U.S. Government.

2.2. Graph Extraction

With the data in the D4M format, we can easily extract a number of graphs based on relationships in the data. The D4M architecture allows for linear algebraic operations on the associative arrays, enabling easy construction of the adjacency matrices of a graph, i.e., a matrix A representing a graph $G = (V, E)$ wherein the entry in the ith row and jth column of A is nonzero only if $(v_i, v_j) \in E$.

Returning to the conference program example, with an associative array $\mathbf{A}$ with rows indexed by titles and columns indexed by authors and sessions. To build a graph in which the vertices are papers and an edge occurs between two vertices if the corresponding papers are in the same session, the procedure is simple. First extract the columns corresponding to sessions, i.e., create a new associative array

$$\widehat{\mathbf{A}} = \mathbf{A}(:, \texttt{'session/0,:,session/z,'}),$$

which covers the range of sessions that begin with alphanumeric values. Now $\widehat{\mathbf{A}}$ is a title-to-session associative array with exactly one nonzero value per row. To create the desired graph, we simply take the inner product of this array with itself, $A = \widehat{\mathbf{A}}^T \widehat{\mathbf{A}}$, and A is the adjacency matrix of the session-cooccurrence graph.

2.3. Graph Analysis

Once the data are in network form, we run graph analysis algorithms to detect interesting or anomalous behavior. In this paper, we focus on the eigenspace analysis techniques outlined in [1, 3, 4]. These algorithms are all based on spectral analysis of the modularity matrix [5], which we interpret as a graph-based residuals matrix. The modularity matrix for an unweighted, undirected graph is given by

$$B = A - \frac{kk^T}{2|E|}.$$

Here A is the adjacency matrix of the graph and k is the degree vector, i.e., a vector in which the ith entry corresponds to the number of edges adjacent to vertex v_i. The term $kk^T/(2|E|)$ represents a first-order fit of the graph to the Chung–Lu random graph model, in which the probability that an edge occurs between two vertices is proportional to the product of their expected degrees [6]. We thus treat B as a matrix of residuals—of the "observed" edges minus the "expected" edges—and analyze it to find anomalies.

The major computation in this analysis is the calculation of the eigenvectors and eigenvalues of B. This is done using `eigs` in Matlab, which uses the implicitly restarted Lanczos method to compute extreme eigenvalues and their corresponding eigenvectors. One issue in doing this computation is that, while a large A is likely very sparse, the corresponding B is dense (indeed, an entry in B will only be zero if an associated vertex is isolated). However, as Newman points out in [5], since B is the sum of a sparse matrix and a rank-1 matrix, we can compute the eigenvectors efficiently without computing B. The function `eigs` can compute eigenvalues and eigenvectors of a function, so by creating $f : \mathbb{R}^{|V|} \to \mathbb{R}^{|V|}$ such that

$$f(x) = Bx = Ax - k\frac{k^T x}{2|E|},$$

we can exploit the form of B for a much less intense computation.

The running time of `eigs` on an adjacency matrix is $O(|E|m + |V|m^2 + m^3)$ per restart, where m is the number of eigenvectors computed. (The number of restarts is dependent upon the gap in magnitude between consecutive eigenvalues of B.) The Lanczos method computes the eigenvalues via iterative matrix-vector multiplication, or in our case iteratively evaluating $f(x)$. Assuming $|V| < |E|$, computing Ax costs $O(|E|)$ operations, and adding the computation and subtraction of $k(k^T x)/(2|E|)$ adds an additional $4|V|$ operations, which does not increase the asymptotic running time. This algorithm, therefore, scales linearly in $|E|$ for a constant m and linearly in $|E|m$ if m grows no faster than the average degree k_{avg}. The space requirement for this method is $O(|V|m)$, which scales linearly in $|E|$ if m is $O(k_{\text{avg}})$.

We are also interested in analyzing directed and dynamic graphs. The modularity matrix for a directed graph is given by

$$\widehat{B} = A - \frac{k_{\text{out}} k_{\text{in}}}{|E|},$$

where k_{out} and k_{in} are, respectively, a column vector of out degrees (the edge counts going *out of* the vertices) and a row vector of in degrees (the edge counts going *into* the vertices). Note that in this case the "expected" term has $|E|$ rather than $2|E|$ in the denominator, since the edges go in only one direction. As in [7], we use the "symmetrized" modularity matrix $B = (\widehat{B} + \widehat{B}^T)/2$. The running time will be greater than for undirected graphs, since it requires multiplication at each iteration by A, A^T, $k_{\text{out}}k_{\text{in}}$ and $k_{\text{in}}^T k_{\text{out}}^T$, but the asymptotic running time will not change, so it will scale similarly.

For dynamic graphs, we consider the filtered modularity matrices over time, as in [4]. Let $B(n)$ be the modularity matrix at discrete time step n. We accumulate the residuals over a time window of ℓ samples by applying a finite impulse response filter h to create an aggregated residuals matrix

$$\tilde{B}(n) = \sum_{i=0}^{\ell-1} h(i)B(n-i), \tag{1}$$

where $h(i)$ is a scalar for all integers $0 \le i < \ell$. Each iteration now requires multiplication by the adjacency matrix and degree vectors at each time step within the window. Letting $E(n)$ be the edge set at time n, this process has running time $O(|V|\ell + \sum_{i=0}^{\ell-1} |E(n-i)|)$. (The vertex set is assumed to be fixed to maintain the matrix dimensions.) This slightly alters the running time of the Lanczos procedure, making it $O(\bar{E}\ell m + |V|\ell m + |V|m^2 + m^3)$, where $\bar{E}$ is the average cardinality of E over the time window. For a fixed window length, this algorithm scales the same as in the static case.

3. DATASET AND GRAPH CONSTRUCTION

The dataset we analyze is the commercially available Thomson Reuters Web of Science® (WoS) database [8]. This dataset is comprised of records, compiled for research purposes, representing scholarly publications of the international scientific community, published between 1900 and present in public commercial and open source journals and conference proceedings. Each record represents an individual document, and fields include document title and type, journal name, author names and institutional affiliations (as provided in publication), cited references, and publication date. There are several interesting dynamic graphs we can extract from this data, including coauthorship graphs, citation graphs, and graphs associated with some notion of document similarity, such as common n-grams.

We obtained a snapshot of the database that contains over 42 million records. The raw data were parsed and inserted into a database that can be accessed via the D4M interface. The data are stored in associative arrays in which the rows correspond to unique

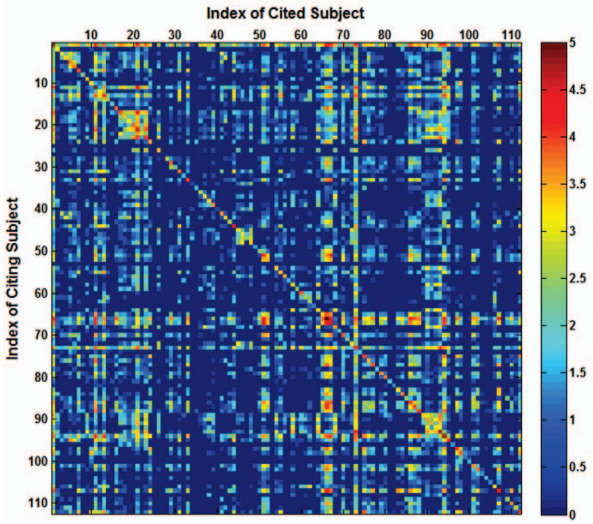

Fig. 1. Subject-to-subject citation counts in the first 50 years of WoS data ($\log_{10}$ scale).

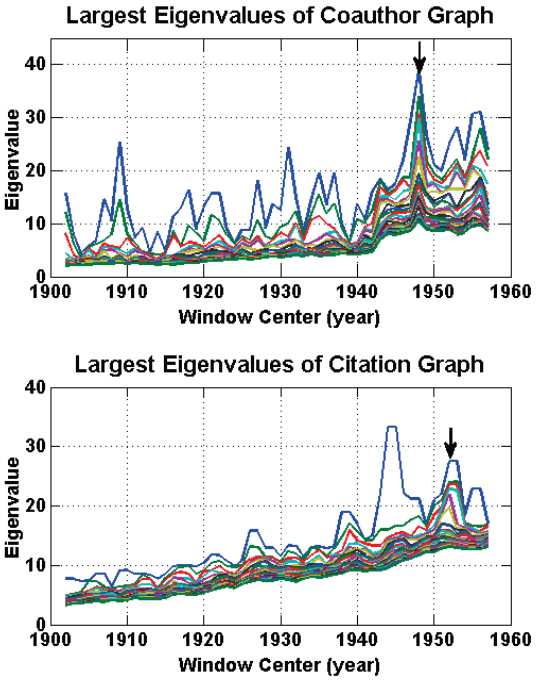

Fig. 2. The largest 30 eigenvalues of the coauthorship graph (top) and citation graph. There is a slow upward trend in both cases with a number of excursions, which in most cases correspond to clutter.

identifiers and the columns are the document metadata. The majority of the data are in a single associative array, with the more text-intensive data, such as titles and abstracts, in a separate table. This data consumes about 300 GB and fits on a single database node.

Of the many potential graphs we can construct from the WoS dataset, we focus on two in this paper: a coauthorship graph and a citation graph. Both graphs are dynamic, and we use a temporal resolution of one year (since in many cases month and day are not available). In the coauthorship graph G_{auth}, the vertices represent authors and two vertices share an edge in $G_{\text{auth}}(n)$ if they coauthor a document published in year n. This is an unweighted graph, although the edges could include weights corresponding to the number of documents coauthored. In the citation graph G_{cite}, the vertices represent documents and a directed edge occurs from $u \in V$ to $v \in V$ in $G_{\text{cite}}(n)$ if u cites v in year n or earlier. The rationale for using a cumulative graph for citations but not authors is that citations are, for the most part, static (i.e., the documents cited are fixed at the time of publication), while authors may change collaborators over time.

These graphs are easily constructed from the associative arrays in which we store the data. To create a coauthorship graph, select the author columns and take the inner product of the resulting associative array with itself. To break up the graph by year, we can first select subsets of rows with nonzero entries in columns corresponding to a certain date range. For citation graphs, the construction process is even more straightforward. Since there are columns corresponding to the identifier for the cited documents, we simply extract the columns for the references.

It is worth noting at this point that, in addition to graphs, we can construct matrices of useful data statistics in a similar fashion. For example, in an anomaly detection problem, we may want to determine the likelihood that an article published in one subject will cite an article in another. To get the subject-to-subject citation counts, we extract a document-to-subject array $\mathbf{A}_{ds}$ and a document-to-document citation array $\mathbf{A}_{\text{cite}}$, and compute $C = \mathbf{A}_{ds}^T \mathbf{A}_{\text{cite}} \mathbf{A}_{ds}$. The subject-to-subject citation count matrix for the first 50 years of WoS records is shown in Fig. 1. The largest value in a given row or column of C tends to fall on the main diagonal, indicating that documents are often most likely to cite (or be cited by) other documents in the same subject area. There are also clusters corresponding to subfields of a topic area; for example, chemistry is split into analytical, applied, and other subfields. Using this information has the potential to increase our modeling ability, providing a better expected value model than the one discussed in Section 2.3 using tools from link prediction and the point process model of [9].

4. DATA ANALYSIS

At the time of this writing, we have extracted G_{auth} and G_{cite} over the course of the first 60 years of WoS records. In the 2 million records in the database over this time period, there are 549,726 unique authors and 4,668,824 documents (including cited documents that are not in the database), so these are the sizes of the vertex sets in the corresponding graphs. We analyze each dynamic graph over a sliding 5-year time window, using a ramp filter (i.e., $h(i)$ in (1) decreases linearly as i increases) to emphasize emerging connectivity. The 30 largest eigenvalues and eigenvectors of $\tilde{B}$ are computed, which, for the (larger) citation graph, takes approximately one hour per time window on a single processor.

The largest eigenvalues of the integrated modularity matrices are shown in Fig. 2. In both graphs, the eigenvalues gradually increase over the course of the 60 years (although more quickly in the citation graph since it is cumulative). There are several points in which the largest eigenvalues deviate substantially from the general trend. We will consider one window from each dynamic graph, as indicated in the figure: the window centered at 1948 for G_{auth} and the window centered at 1952 for G_{cite}. In both of these windows, several of the

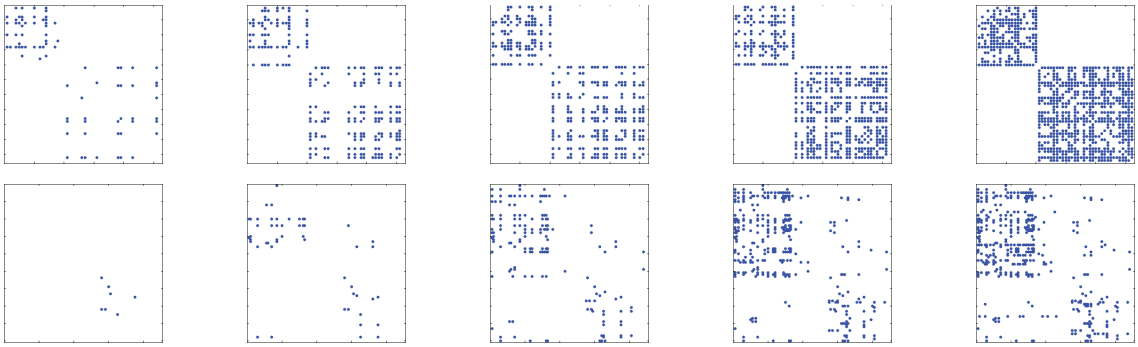

Fig. 3. Emerging clusters in the WoS graphs. Adjacency matrices are shown for subsets of the coauthorship graph (top row) and the citation graph (bottom row) over 5-year time windows (increasing year from left to right). In both cases, tightly connected clusters emerge over time.

eigenvalues are significantly larger than eigenvalues of similar rank in earlier and later windows.

4.1. Clutter

Upon deeper inspection, we find that many of the vertices that stand out in the space of largest modularity are somewhat uninteresting, which we see as analogous to clutter in radar processing. In G_{auth}, several of the eigenvectors with large eigenvalues are aligned with large cliques in which no authors had previously collaborated. These large cliques (on the order of 50 vertices) often occur for documents of type "discussion", and could be easily detected by finding documents with large author lists.

In the citation graph we see a different kind of clutter. The residuals space of G_{cite} is often dominated by review articles that cite many hundreds (sometimes thousands) of documents. Again, we do not really benefit from analyzing the data as a dynamic graph, as these vertices could be found by simply looking for documents with long lists of references.

4.2. Emerging Clusters

Looking under the clutter, however, we see some interesting behavior involving emerging clusters of nodes. Looking in the space of eigenvectors that do not correspond to clutter behavior, we find a few vertex subsets that stand out from the rest of the graph. Sparsity patterns of the adjacency matrices corresponding to these subgraphs are shown in Fig. 3. In G_{auth}, we see two sets of authors (publishing mostly in medical journals), each of which gradually increases the number of connections within the subset over the window. These subgraphs are smaller than the large cliques that appear suddenly in the same window: one has 20 vertices and the other has 32, and neither is ever a clique. However, our temporal integration technique emphasizes the increasing connectivity over time and brings these subgraphs into the space of eigenvectors with larger eigenvalues.

We see similar behavior in G_{cite}. In this case, as time passes, we see two subsets of documents accumulating a significant amount of internal citation, with some citation between the two subsets. Most documents in these subsets are in biochemistry and microbiology, with a few in medicine and other areas, and largely focus on metabolic properties of various acids and proteins. Again, the emerging connectivity is emphasized by the ramp filter and these relatively small subsets are brought into the space of large residuals.

5. SUMMARY

In this paper, we outline an architecture for analyzing large graph data. The architecture is supported by the D4M framework, allowing easy and intuitive construction of graphs from databases, and uses an efficient eigenspace analysis technique for signal-processing-based analysis on the constructed graphs. We analyze two large, dynamic graphs derived from the Web of Science database (one with over 500 thousand and one with over 4 million nodes), and find different types of clutter in the different graphs, as well as small, emerging clusters in the eigenspace of graph residuals. Future work will include developing automated methods to filter away clutter in the graph data, incorporating metadata into our models of graph residuals, and analyzing multi-graphs in which different types of edges correspond to different relationships.

6. REFERENCES

[1] B. A. Miller, N. T. Bliss, and P. J. Wolfe, "Toward signal processing theory for graphs and non-Euclidean data," in *Proc. ICASSP*, 2010, pp. 5414–5417.

[2] J. Kepner, "Massive database analysis on the cloud with D4M," in *Proc. HPEC Workshop*, 2011.

[3] B. A. Miller, N. T. Bliss, and P. J. Wolfe, "Subgraph detection using eigenvector L1 norms," in *Advances in Neural Inform. Process. Syst. 23*, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., 2010, pp. 1633–1641.

[4] B. A. Miller, M. S. Beard, and N. T. Bliss, "Matched filtering for subgraph detection in dynamic networks," in *Proc. IEEE Statistical Signal Process. Workshop*, 2011.

[5] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E*, vol. 74, no. 3, 2006.

[6] F. Chung, L. Lu, and V. Vu, "The spectra of random graphs with given expected degrees," *PNAS*, vol. 100, no. 11, pp. 6313–6318, 2003.

[7] E. A. Leicht and M. E. J. Newman, "Community structure in directed networks," *Physics Review Letters*, vol. 100, 2008.

[8] "Thomson Reuters Web of Science," http://thomsonreuters.com/products_services/science/science_products/a-z/web_of_science.

[9] P. O. Perry and P. J. Wolfe, "Point process modeling for directed interaction networks," 2010, http://arxiv.org/abs/1011.1703.