ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 2, Issue 12, December 2017

Visual Analysis for Data Exploration – Wavelet

[1] J. Maria Merceline Vijila, [2] R. Ame Rayan
[1][2] Assistant Professor, Department of Computer Science.

Abstract:-- **In image and signal processing, the conventional wavelet transform is used. In this,the signal is decomposed into a combination of known signals. To analyse the contribution of an individual, the original signal's behaviour can be inferred. In this article, an overview of the extension of this theory into graph domains ispresented as an introductory by the author's. In this we are about to review the graph Fourier transform and graph wavelet transforms. These transforms are based on dictionaries of graph spectral filters, namely, spectral graph wavelet transforms. By this we present the main features of the graph wavelet transforms using real and synthetic data. The challenging problem that has been faced is to visualize time-varying data defined on the nodes of a graph. As a result we show our approach using synthetic as well as a real data set.**

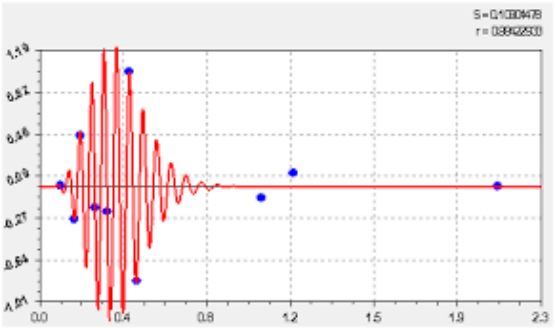
INTRODUCTION

The theory of graph signal processing has emerged as a main alternative to analyze signals defined on the vertices of graphs . It focuses on adapting tools and operators, such as Fourier and Wavelet transforms, to the context of graphs, opening a multitude of possibilities to the exploration and understanding of static and time-varying phenomena in irregular domains.

Similarly to the case of regular domains, most of the analysis derived from Graph Fourier (GFT) and Graph Wavelet Transforms (GWT), relies on the processing of coefficients associated with certain basis functions defined on the vertices of a graph .

However, the irregular nature of graph domains makes the coefficients sensitive not only to the properties of the signal under analysis, but also to the topology of the graph. In fact, the topology of the graph directly impacts how the frequencies are spread in the graph Fourier domain, making the design of filters a more intricate task . Filters designed without considering the distribution of frequencies may lead to wrong analysis and processing results. The issue of properly designing filters to operate with GFT and GWT becomes even more challenging for inexperienced users that are not usually aware of the nuances involved in the transforms. However, the sensitivity of the GFT and GWT to signal variation across irregular domains is an evident and noteworthy characteristic of these transforms, which can be used to understand phenomena related to daily life. For example, the street map of a city can be represented as a graph and the way the data under analysis varies across streets, from corner to corner, may reveal important aspects of city's dynamics.

In this paper we discuss some aspects of the GFT and GWT, providing examples of how the signal and the topology of the graph impact the computation of coefficients. Our goal is not to provide a rigorous treatment towards understanding the effect of the topology on GFT and GWT coefficients, but to shed some light on it to help inexperienced users in the use of these tools. In fact, the clear understanding of the synergy among GFT and GWT coefficients, graph topology, and the signal under analysis is still an open issue, which we consider out We present a set of controlled experiments, using real and synthetic data, to illustrate how the topology and signal can affect the result of GFT and GWT. We bring some intuition from the conventional Fourier and Wavelet theory to the more complex scenario of graph domains, relying on simple, but illustrative, examples showing the dissociation between spectral distribution and filter design in this context. We conclude this article with a real application that reveals the power of GFT and GWT in uncovering patterns and interesting phenomena hidden in the data.



2. GRAPH WAVELET TRANSFORMS

E ! R that associates a non-negative scalar to each edge of G. We assume that G is connected, for every pair of nodes, there is a path (sequence of edges such that any two sequential edges contain a node in common) connecting them.

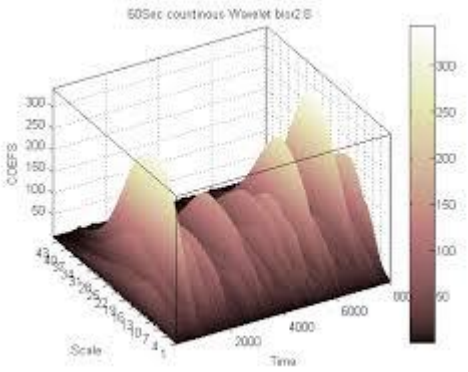
The weighted adjacency matrix of G is the matrix A = (ai j), where ai j = w(ti; t j) if (ti; t j) 2 E and ai j = 0 otherwise. This matrix is used to define the (non-normalized) graph Laplacian, given by L = D□A, where D = diag(d1;d2; : : :

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 2, Issue 12, December 2017

;dn) is a diagonal matrix with entries di = åj ai j. The graph Laplacian is a real, symmetric, and semi-positive definite matrix, thus it has a complete set of orthonormal eigenvectors u`, with corresponding non-negative real eigenvalues l`, ` = 1;2; : : : ;n, which we consider ordered in non-decreasing order 0 = l1 < l2 _ ___ _ ln. A signal defined on the nodes of G f : V ! R is a function that associates a real value f (ti) to each node ti 2V. The graph Laplacian eigenvectors u` : V !R can be interpreted as signals on the graph G. The GFT provides a global description of the signal behavior on the graph. For example, let f0 be a signal defined as a combination of low and high frequency The GFT of f0, indicates the presence of low and high frequencies in the signal, suggesting that the signal has both smooth and abrupt variation. The yellow curve illustrates the magnitude of the conventional Discrete Fourier Transform (DFT) of f0 when defined on the real line.

A graph spectral filter, or kernel, ˆ g : L ! R is a function defined in the spectral domain that associates a scalar value ˆ g(l`) to each eigenvalue l` 2 L. The GFT ˆ f : L ! R can be seen as a particular instance of a graph spectral filter. A dictionary f ˆ gmgm=1;2;:::;M is a set of graph spectral filters ˆ gm : L ! R, m = 1;2; : : : ;M, where M is the number of kernels in the dictionary. Given a signal f and a dictionary f ˆ gmgm=1;2;:::;M, the Graph Wavelet Transform (GWT) Wf : f1;2; : : : ;Mg_V !R is defined as: Wf (m; t j) = nå`=1 ˆ gm(l`) bf (l`)u`(t j): (3). By varying m while keeping t j fixed, we obtain M wavelet coefficients associated to t j. Moreover, the product ˆ gm(l`) bf (l`), on the right of Equation 3, shows that each coefficientWf (m; t j) is obtained by modulating bf (l`) by the kernel ˆ gm(l`). Interpreting ˆ gm as a band-pass filter, where small values of m correspond to low-pass and larger values to high-pass filters, the coefficients encode the behavior of the signal in each node. The coefficients corresponding to low frequencies (m _ 4) in node t15 are larger than high frequency coefficients, reflecting the fact that the signal in node t15 is smoother. In contrast, larger coefficients appear in high frequencies, indicating that the signal oscillates more abruptly around in node



3. UNDERSTANDING THE INTERPLAY AMONG SPECTRUM, TOPOLOGY, AND COEFFICIENTS

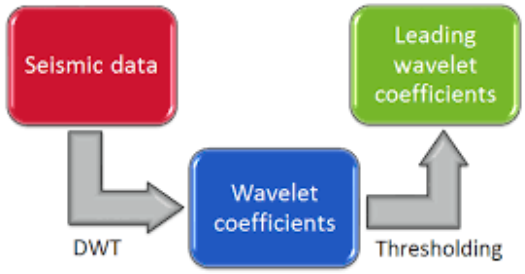
The goal of this paper is to develop zenvisage, a visual analytics system that can automate the search for desired visual patterns. Our key insight in developing zenvisage is that the data exploration needs in all of these scenarios can be captured within a common set of operations on collections of visualizations. These operations include: composing collections of visualizations, filtering visualizations, based on some conditions, comparing visualizations, and sorting them based on some condition. The conditions include similarity or dissimilarity to a specific pattern, “typical” or anomalous behavior, or the ability to provide explanatory or discriminatory power. These operations and conditions form the kernel of a new data exploration language, ZQL ("zee-quel"), that forms the foundation upon which zenvisage is built.

We encountered many challenges in building the zenvisage visual analytics platform, a substantial advance over the manually-intensive visualization tools like Tableau and Spotfire; these tools enable the examination of one visualization at a time, without the ability to automatically identify relevant visualizations from a collection of visualizations. First, there were many challenges in developing ZQL, the underlying query language for zenvisage. Unlike relational query languages that operate directly on data, ZQL operates on collections of visualizations, which are themselves aggregate queries on data. Thus, in a sense ZQL is a query language that operates on other queries a a first class citizen. This leads to a number of challenges that are not addressed in a relational query language context. For example, we had to develop a natural way to users to specify a collection of visualizations to operate on, without having to explicitly list them; even though the criteria on which the visualizations were compared varied widely, we had to develop a small number of general mechanisms that capture all of these criteria; often, the visualizations that we operated on had to be modified in various ways—e.g., we might be interested in visualizing the sales of a product whose profits have been dropping—composing these visualizations from existing ones is not straightforward; and lastly, drilling down into specific visualizations from a collection also required special care. Our ZQL language is a synthesis of desiderata after discussions with data scientists from a variety of domains, and has been under development for the past two years. To further show that ZQL is complete under a new visual exploration algebra that we develop, involved additional challenges. Second, in terms of front-end development, zenvisage, being an interactive analytics tool, needs to

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 2, Issue 12, December 2017

support the ability for users to interactively specify ZQL queries—specifically, interactive shortcuts for commonly used ZQL queries, as well as the ability to pose extended ZQL queries for more complex needs. Identifying common interaction "idioms" for these needs took many months. Third, an important challenge in building zenvisage is the backend that supports the execution of ZQL. A single ZQL query can lead to the generation of 10000s of visualizations—executing each one independently as an aggregate query, would take several hours, rendering the tool somewhat useless. (As it turns out, this time would be what an analyst aiming to discover the same pattern would have to spend with present visualization tools, so the naive automation may still help reducing the amount of manual effort.) zenvisage's query optimizer operates as a wrapper over any traditional relational database system. This query optimizer compiles ZQL queries down to a directed acyclic graph of operations on collections of visualizations, followed with the optimizer using a combination of intelligent speculation and combination, to issue queries to the underlying database. We also demonstrate that the underlying problem is NP-HARD. Our query optimizer leads to substantial improvements over the naive schemes adopted within relational database systems for multi-query optimization. Related Work. There are a number of tools one could use for interactive analysis; here, we briefly describe why those tools are inadequate for the important need of automating the search for desired visual insights. We describe related work in detail in Section 8. To start, visualization tools like Tableau and Spotfire only generate and provide one visualization at a time, while zenvisage analyzes collections of visualizations at a time, and identifies relevant ones from that collection—making it substantially more powerful. While we do use relational database systems as a computation layer, it is cumbersome to near-impossible to express these user needs in SQL. As an example, finding visualizations of solvents for whom a given property follows a roughly increasing trend is impossible to write within native SQL, and would require custom UDFs—these UDFs would need to be hand-written for every ZQL query. Similarly, finding visualizations of keywords where CTR over time in Asia is behaving unusually with respect to other keywords is challenging to write within SQL. For the small space of queries where it is possible to write the queries within SQL these queries require non-standard constructs, and are both complex and cumbersome to write, even for expert SQL users, and are optimized very poorly (see Section 8). It is also much more natural for endusers to operate directly on visualizations than on data. Indeed, users who have never programmed or written SQL before find it easy to understand and write a subset of ZQL queries, as we will show subsequently. Statistical, data mining, and machine learning certainly provide functionality beyond zenvisage in supporting prediction and statistics; these functionalities are exposed as "one-click" algorithms that can be applied on data. However, no functionality is provided for searching for desired patterns; no querying functionality beyond the one-click algorithms, and no optimization. To use such tools for ZQL, many lines of code and hand-optimization is needed. As such, these tools are beyond the reach of novice data scientists who simply want to explore and visualize their datasets.



4. THE CONCEPT OF VISUALIZATIONS.

We start by defining the notion of a visualization. We use a sample visualization in Figure 1 to guide our discussion, Of course, different visual analysis tasks may require different types of visualizations (instead of bar charts, we may want scatter plots or trend lines), but across all types a visualization is defined by the following five main components: (i) the x-axis attribute, (ii) the y-axis attribute, (iii) the subset of data used, (iv) the type of visualization (e.g., bar chart, scatter plot), and (v) the binning and aggregation functions for the x- and y- axes. Name X Y Z Viz *f1 'year' 'sales' 'product'.'chair' bar.(y=agg('sum')) Table 1: Query for the bar chart of sales over year for the product chair. Name X Y Z Viz *f1 'year' 'sales' 'product'.* bar.(y=agg('sum')) Table 2: Query for the bar chart of sales over year for each product. Visualization collections in ZQL: ZQL has four columns to support the specification of visualizations that the five aforementioned components map into: (i) X, (ii) Y, (iii) Z, and (iv) Viz. Table 1 gives an example of a valid ZQL query that uses these columns to specify a bar chart visualization of overall sales over the years for the product chair (i.e., the visualization in Figure 1)— ignore the Name column for now. The details for each of these columns are presented subsequently. In short, the x axis (X) is the attribute year, the y axis (Y) is the attribute sales, and the subset of data (Z) is the product chair, while the type of visualization is a bar chart (bar), and the binning and aggregation functions indicate that the y axis is an aggregate (agg) — the sum of sales. In addition to specifying a single visualization, users may often want to retrieve multiple visualizations. ZQL supports this in two ways. Users

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 2, Issue 12, December 2017

may use multiple rows, and specify one visualization per row. The user may also specify a collection of visualizations in a single row by iterating over a collection of values for one of the X, Y, Z, and Viz columns. Table 2 gives an example of how one may iterate over all products (using the notation * to indicate that the attribute product can take on all values), returning a separate sales bar chart for each product. High-level structure of ZQL. Starting from these two examples, we can now move onto the general structure of ZQL queries. Overall, each ZQL query consists of multiple rows, where each row operates on collections of visualizations. Each row contains three sets of columns, as depicted in Table 3: (i) the first column corresponds to an identifier for a visualization collection, (ii) the second set of columns defines a visualization collection, while (iii) the last column corresponds to some operation on the visualization collection. All columns can be left empty if needed (in such cases, to save space, for convenience, we do not display these columns in our paper). For example, the last column may be empty if no operation is to be performed, like it was in Table 1 and 2. We have already discussed (ii); now we will briefly discuss (i) and (iii), corresponding to Name and Process respectively. Identifiers and operations in ZQL. The Process column allows the user to operate on the defined collections of visualizations, applying high-level filtering, sorting, and comparison. The Name column provides a way to label and combine specified collections of visualizations, so users may refer to them in the Process column. Thus, by repeatedly using the X, Y, Z, and Viz columns to compose visualizations and the Process column to process those visualizations, the user is able derive the exact set of visualizations she is looking for. Note that the result of a ZQL query is the data used to generate visualizations. The zenvisage front-end then uses this data to render the visualizations for the user to peruse.

In the Cartesian graph, the coefficients derived from GFT and GWT
capture spatial and temporal behavior of the signal, making their interpretation
more intricate than in the static case. In order to gain intuition, we design a Cartesian graph G_H where G is as depicted in the leftmost panel and H is a path graph with twenty nodes; G_H comprises twenty time slices of a time-varying signal defined on its nodes. We also craft four different time-varying signals on G_H denoted as fi; i = 1;2;3; 4. The signal f1 is defined with a spike (value equal 1) in the node tai10 of G_H, vanishing in the remaining nodes. The nodes ta and i10 are enhanced in the leftmost panel. In other words, f1 has a spike in the node ta in time slice i10. The signal f2 is defined as a “spatial” Gaussian centered in tai10, assigning values for nodes in the neighborhood of ta in time slice 10 and zero in other time slices. The signal f3 is defined as a “temporal” Gaussian centered in tai10, assigning values for nodes tai j; j = 1; : : : ;20 and zero elsewhere. Finally, f4 is the combination of a Gaussian over time centered at tai10 and Gaussians in all time
slices with spikes in tai j; j = 1; : : : ; 20.

5. PROOF OF VISUAL EXPLORATION COMPLETENESS

We now attempt to quantify the expressiveness of ZQL within the context of visual exploration algebra and the two functional primtives T and D. More formally, we prove the following theorem: THEOREM 5.1. Given well-defined functional primitives T and D, ZQL is visual exploration complete with respect to T and D: V ECT,D(ZQL) is true. Our proof for this theorem involves two major steps: Step 1. We show that a visualization collection in ZQL has as much expressive power as a visual group of visual exploration algebra, and therefore a visualization collection in ZQL serves as an appropriate proxy of a visual group in visual exploration algebra. Step 2. For each operator in visual exploration algebra, we show that there exists a ZQL query which takes in visualization collection semantically equivalent to the visual group operands and produces visualization collection semantically equivalent to the resultant visual group.

6. CONCLUSION

We have shown the sensitivity of the GFT and GWT to signal variation across irregular domains. These transforms gracefully combine topology with the data under analysis, such that they can be used to understand phenomena related to daily life. For instance, a homogeneous number of taxi pick-ups across the street map of a city suggests a large event happening in the city, while a more heterogeneous pattern can lead us to influential localities. Further, the temporal evolution of data can be incorporated in the process in order to use GWT properties in spatio-temporal analyses.

Graph signal processing is still in its earlier days and much has to be done in order to make it accessible to experienced users from different fields. As we have shown in this article, many issues are still open, including how to change the edge weights to mitigate the impact of the topology in the graph Fourier coefficients and graph wavelet coefficients. Other interesting problem is how to design filters with particular properties such as boundary detection filters, which could be useful to perform unconventional clustering in spatio-temporal data.

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 2, Issue 12, December 2017

REFERENCES

[1] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. Applied and Computational Harmonic Analysis, 30(2):129–150, 2011.

[2] D. M. Mohan, M. T. Asif, N. Mitrovic, J. Dauwels, and P. Jaillet. Wavelets on graphs with application to transportation networks. In IEEE International Conference on Intelligent Transportation Systems, pages 1707– 1712. IEEE, 2014.

[3] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high- dimensional data analysis to networks and other irregular domains. IEEE Signal Processing Magazine, 30(3):83–98, 2013.

[4] D. I. Shuman, C. Wiesmeyr, N. Holighaus, and P. Vandergheynst. Spectrum-adapted tight graph wavelet and vertex-frequency frames. IEEE Transactions on Signal Processing, 63(16):4223–4235, 2015.

[5] N. Tremblay and P. Borgnat. Graph wavelets for multiscale community mining. IEEE Transactions on Signal Processing, 62(20):5227–5239, 2014.

[6] P. Valdivia, F. Dias, F. Petronetto, C. T. Silva, and L. Nonato. Waveletbased visualization of time-varying data on graphs. In IEEE Conference on Visual Analytics Science and Technology. IEEE, 2015.