Windowing Methods for Graph Signal Localization

Miloš Daković, Ljubiša Stanković
University of Montenegro
Podgorica, Montenegro
Email: {milos, ljubisa}@ac.me

Ervin Sejdić
University of Pittsburgh
Pittsburgh, Pennsylvania, USA
Email: esejdic@ieee.org

***Abstract*—In this paper we considered windows used for local vertex spectrum analysis of graph signals. In addition to a review of the convolution based windowing method, two methods based on the vertex neighborhood are presented. They are based on the graph path lengths. In the first one the number of edges in a path determine window size, while the edge weights are taken into account in the second method. Signal localization is performed by using these window functions. Windowing methods are used for signal local vertex spectrum calculation with a test signal. Norm one based concentration measure is used for comparison.**

***Index Terms*—graph signal processing, windows, local vertex spectrum, graph Laplacian**

I. Introduction

Graph signal processing is a challenging, but rapidly developing, topic. Many real world signals can be considered as graph signals, i.e., signals defined on a graph. Basic and advanced graph signal processing techniques are presented in [1]–[4], and some of its applications in biomedical systems [5], [6] and big data analysis [7] are best examples of its real-world potential.

In the case of large signals (graphs), we may not be interested in the analysis of the entire signal, but rather interested in its local behavior. A localized signal behavior can be examined via window functions. An exemplary analysis is signal averaging in a local neighborhood. This kind of processing correspond to low-pass filtering in the classical time-domain signal analysis. Another example could be a classical time-frequency analysis [8], [9] where we consider a local signal spectrum. In both examples, window functions are used in order to perform signal localization in time. Window functions are often symmetric, with a single maximum value at a considered time instant. Window functions can be easily shift in time in order to analyze a signal behavior at arbitrary time instants.

This concept of signal localization by using window functions can be extended to signals defined on graphs [6], [10]–[14]. The extension is not straightforward since a simple operation like time shifting cannot be easily defined in a graph signal domain. Several solution approaches for this problem are defined.

A common approach is to utilize the signal spectrum to obtain window functions for each graph vertex [4]. Another possibility is to define a window support as local neighborhood for each vertex [14]. The localization window is defined by a set of vertices that contain the current vertex n and all vertices that are close to the vertex n. As in the classical signal analysis, a window should be narrow enough in order to provide good localization of the signal properties but wide enough to produce high resolution.

In this paper, we will focus on the localized vertex spectrum of a graph signal. The basic concepts are presented in Section II. Localization methods are analyzed in Section III. The obtained results are shown in Section IV.

II. Graph signals

Consider a weighted undirected simple graph with N vertices where edge weights are denoted by $w_{nm} > 0$ for an edge that connects a vertex n with a vertex m and $w_{nm} = 0$ if vertices n and m are not connected with an edge.

Edge weights could be defined in many ways. If the considered graph correspond to the Euclidean network, where vertices are points in a plane, edge weights could be defined as the Euclidean distance between corresponding vertices. It is common that higher w_{nm} values indicate strong connections and low w_{nm} values indicates weak connections between considered vertices. Then, we can use a decreasing function of a distance as edge weights. A common approach is to define weights as $w_{nm} = \exp(-\tau r_{nm})$ or $w_{nm} = \exp(-\tau r_{nm}^2)$ where r_{nm} is the distance between considered vertices and $\tau > 0$ is a constant. If a considered graph corresponds to the resistive electrical circuit, the edge weights could be defined as conductances $w_{nm} = 1/R_{nm}$, where R_{nm} is the resistance of the considered edge [14], [15]. These examples are just two possibilities for defining the edge weights. We will assume that edge weights are a-priori defined, in accordance with the considered application.

Edge weights are often represented in a matrix form

$$\mathbf{W} = \begin{bmatrix} 0 & w_{12} & w_{13} & \cdots & w_{1N} \\ w_{21} & 0 & w_{23} & \cdots & w_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & w_{N3} & \cdots & 0 \end{bmatrix} \tag{1}$$

The diagonal matrix elements of $\mathbf{W}$ are zeros, since the graph is simple (without loops). The weighting matrix $\mathbf{W}$ is a symmetric matrix since the considered graph is undirected.

Signal $x(n)$, defined at each graph vertex n, is called graph signal. Signal samples $x(n)$ can be arranged in a $N \times 1$ vector $\mathbf{x} = [x(1), x(2), \ldots, x(N)]^T$.

The graph Laplacian plays an important role in graph signal processing. It is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}, \tag{2}$$

where $\mathbf{D}$ is a diagonal degree matrix with $d_{nn} = \sum_{m=1}^{N} w_{nm}$ on the main diagonal.

The eigenvalue decomposition of the Laplacian matrix reads

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T, \tag{3}$$

where $\mathbf{U}$ is a matrix of eigenvectors (eigenvector $\mathbf{u}_k$ is the kth column of the matrix $\mathbf{U}$), and $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues λ_k on the main diagonal.

The spectrum of a graph signal is defined as $\mathbf{X} = \mathbf{U}^T\mathbf{x}$, where the vector $\mathbf{X}$ contain spectral coefficients connected to the kth eigenvalue and the corresponding eigenvector

$$X(\lambda_k) = \mathbf{u}_k^T\mathbf{x} = \sum_{n=1}^{N} x(n)u_k(n). \tag{4}$$

The inverse transformation is obtained as $\mathbf{x} = \mathbf{U}\mathbf{X}$, with

$$x(n) = \sum_{k=1}^{N} X(\lambda_k)u_k(n). \tag{5}$$

The localized vertex spectrum (LVS) on a graph can be calculated as the spectrum of a signal $x(n)$ multiplied by an appropriate window function $h_m(n)$

$$LVS(m,\lambda_k) = \sum_{n=1}^{N} x(n)h_m(n)\ u_k(n). \tag{6}$$

It is assumed that the window function $h_m(n)$ should be such that it localizes the signal content around the vertex m.

The inversion, i.e. obtaining a signal $x(n)$ from its local spectrum $LVS(m,\lambda_k)$ is performed by an inverse transformation and summation over all vertices

$$x(n) = \frac{1}{\sum_{m=1}^{N} h_m(n)} \sum_{m=1}^{M}\sum_{k=1}^{N} LVS(m,\lambda_k)u_k(n) \tag{7}$$

For $h_m(n) = 1$, the localized vertex spectrum is the equal to the standard spectrum $LVS(m,\lambda_k) = X(\lambda_k)$ for each m, i.e., no vertex localization is performed. If $h_m(m) = 1$ and $h_m(n) = 0$ for $n \neq m$, the localized vertex spectrum is equal to the signal for each k and we do not have any spectral resolution. In the next section, we will show how to create a set of window functions $h_m(n)$ that provide good vertex localization and good spectral resolution.

III. Localization windows

We will present three methods for defining window functions $h_m(n)$ on a given graph. The localization windows will be illustrated on a simple graph with $N = 8$ vertices.

A. Method 1

This method is based on the spectral domain localization windows [10], [11]. In order to define the localized spectrum, consider two signals $x(n)$ and $h(n)$ on a graph. The signal $h(n)$ will be used to localize the spectral characteristics of $x(n)$. Spectra of signals are given by $X(\lambda_k)$ and $H(\lambda_k)$. For two signals on a graph and their spectra Parseval's theorem holds

$$\sum_{k=1}^{N} x(n)h(n) = \sum_{k=1}^{N} X(\lambda_k)H(\lambda_k). \tag{8}$$

A shift of a signal on a graph can not be extended in a direct way from the classical signal processing theory. To achieve this, a generalized convolution operator on a graph is defined. It is assumed that the spectrum of a convolution $y(n) = x(n) * h(n)$ on a graph is equal to the product of signal spectra

$$Y(\lambda_k) = X(\lambda_k)H(\lambda_k). \tag{9}$$

The generalized convolution operation $x(n) * h(n)$ is equal to the inverse transform of $Y(\lambda_k)$,

$$\begin{aligned} y(n) = x(n) * h(n) &= \sum_{k=1}^{N} Y(\lambda_k)u_k(n) \\ &= \sum_{k=1}^{N} X(\lambda_k)H(\lambda_k)u_k(n). \end{aligned}$$

Let us consider the delta function located at a graph vertex m and its spectrum. The delta function at the vertex m is defined as

$$\delta_m(n) = \begin{cases} 1 & \text{for } n = m \\ 0 & \text{for } n \neq m, \end{cases} \tag{10}$$

and the corresponding spectrum is given by

$$\Delta(\lambda_k) = \sum_{n=1}^{N} \delta_m(n)u_k(n) = u_k(m). \tag{11}$$

A shift on a graph can be defined as generalized convolution of the signal $h(n)$ and the delta function located at the vertex m. Here, we use $h_m(n)$ to denote a shifted window. The window localized at the vertex m is equal to

$$h_m(n) = h(n) * \delta_m(n) = \sum_{k=1}^{N} H(\lambda_k)u_k(m)u_k(n). \tag{12}$$

The window basic function is defined in a spectral domain, for example, as $H(\lambda_k) = C\exp(-\lambda_k\tau)$, where C is the window amplitude and $\tau > 0$ is the constant that determines the window width. Windows obtained using this method are presented in Fig. 1.

B. Method 2

Another approach is to define a neighborhood of the considered vertex and assume that the window function is zero outside the considered neighborhood.

The vertex m neighborhood can be defined as a set of vertices $\mathbb{J}_m$ where for each $n \in \mathbb{J}_m$ exists a path, no longer than B from the vertex m to the vertex n. B determines the window support. The length of the shortest path between vertices n and m will be denoted with p_{nm}. Note that $p_{nm} = 1$ if there is an edge (a path of length 1) between vertices n ans

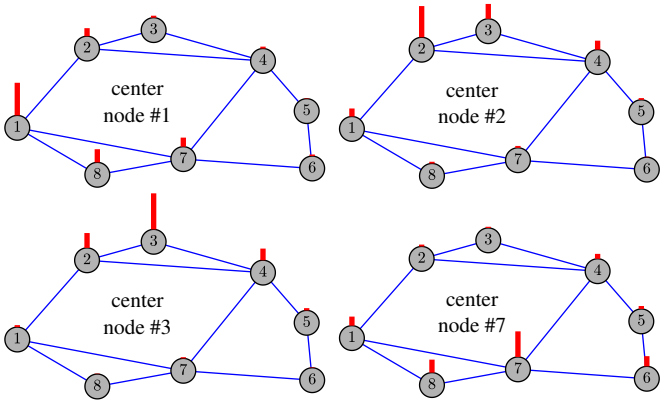

Fig. 1. Method 1: Windows centered at vertices 1, 2, 3 and 7. Window values at each vertex are presented with vertical red bars. Windows are obtained by using shift operation defined by generalized convolution.

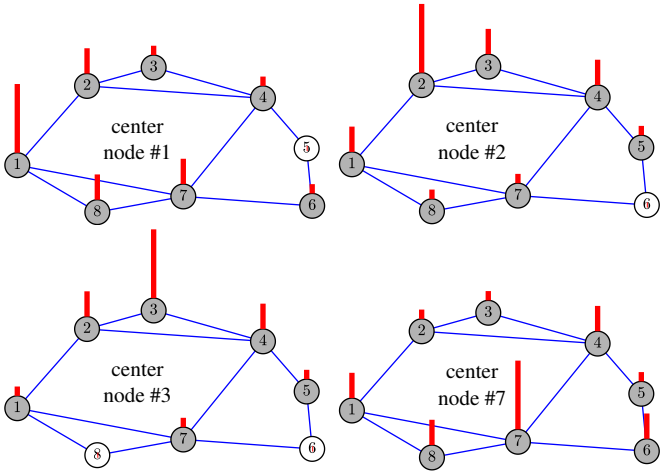

Fig. 2. Method 2: Windows centered at vertices 1, 2, 3 and 7. Window values at each vertex are presented with vertical red bars. For each vertex window support is obtained as set of vertices that are reachable by at most two steps form the considered vertex. Vertices that belongs to window support are shaded.

m. We will assume that $m \in \mathbb{J}_m$ with $p_{mm} = 0$. Window functions can be defined as

$$h_m(n) = \begin{cases} 0 & \text{for } n \notin \mathbb{J}_m \\ 1 & \text{for } n \in \mathbb{J}_m \end{cases} \tag{13}$$

This window correspond to the rectangular window in classical signal analysis. Better localization can be obtained by using windows that decay when p_{nm} increases, for example we can define window values as

$$h_m(n) = \begin{cases} 0 & \text{for } n \notin \mathbb{J}_m \\ \exp(-\tau p_{mn}) & \text{for } n \in \mathbb{J}_m \end{cases} \tag{14}$$

The window function defined by (14) assumes discrete values from the set $\{0, 1, e^{-\tau}, e^{-2\tau}, \ldots, e^{-B\tau}\}$. Sample windows are depicted in Fig 2.

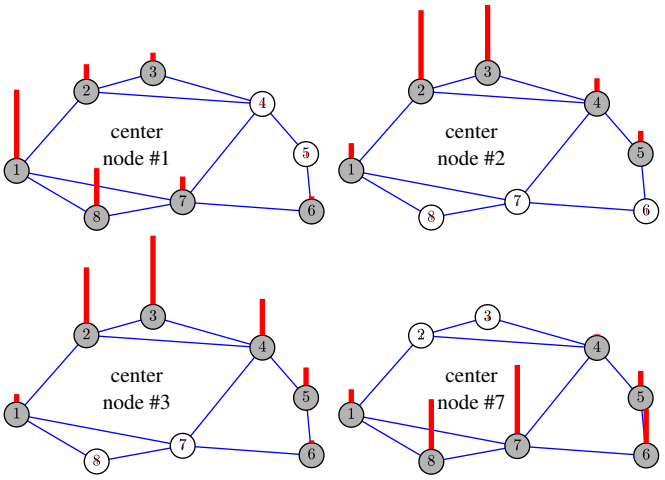

Fig. 3. Method 3: Windows centered at vertices 1, 2, 3 and 7. Window values at each vertex are presented with vertical red bars. Window support and values are obtained by thresholding optimal path weight between considered vertex and remaining vertices. Vertices that belongs to window support are shaded.

C. Method 3

Consider vertices n and m, and all possible paths $\mathcal{P}_n^m$ from n to m. The path $\mathcal{P}_n^m$ is an ordered set of edges (k, l) where the ending vertex of each edge is the starting vertex of the next edge. For each path, we can define a path weight as the product of corresponding edge weights w_{kl}. The maximal product can be used as a decision criterion whether a considered vertex n should or should not be included in the localization window centered at the vertex m. So, the vertex n belongs to the localization window support with the generalized width h_T if

$$\max_{\mathcal{P}_n^m} \prod_{(k,l)\in\mathcal{P}_n^m} w_{kl} = v_{nm} \geq h_T, \tag{15}$$

where $\mathcal{P}_n^m$ is a path from the vertex n to the vertex m.

If the edge weights are $0 \leq w_{kl} \leq 1$ then $0 \leq p_{nm} \leq 1$. A generalized window width h_T ranges from 0 to 1 where 0 produces the widest possible window and 1 gives the narrowest window.

Note that, as in the previous case, we can define a window support as $\mathbb{J}_m$ and the corresponding window functions as

$$h_m(n) = \begin{cases} 0 & \text{for } n \notin \mathbb{J}_m \\ v_{mn} - h_T & \text{for } n \in \mathbb{J}_m \end{cases} \tag{16}$$

By setting $h_m(n) = 1$ for $n \in \mathbb{J}_m$, the window analog to the classical rectangular window is obtained. For better localization, we should define window values $h_m(n)$ such that they favor closer and decrease farther vertex samples. Windows obtained by using this method are illustrated in Fig. 3.

Note that for the windowed signal $x(n)h_m(n)$ only $M \leq N$ samples are nonzero. It may be considered as a classical zero padded signal. It means that for the reconstruction of this signal we only need M spectral coefficients $LVS_x(n, \lambda_k)$ for M different values of λ_k. The remaining coefficients can be

calculated from the system of equations obtained by using the fact that $x(n)h_m(n) = 0$ outside the window support.

In order to visualize a local spectral content, we should order vertices in the corresponding graph. This ordering is not unique and one possible way is to define the order according to the values of low order eigenvectors of the Laplacian. We can, for example, try to minimize the number of zerro crossing in the low order eigenvectors by an appropriate vertices reordering. This can be achieved by reordering vertices such that elements of $b(n)$ are nondecreasing where $b(n)$ is defined as

$$b(n) = \sum_{k=2}^{K} (1 + \text{sign}(u_k(n)))2^{-k} \tag{17}$$

where K is the number of considered eigenvectors. Note that for $k = 1$, and the connected graph, we have $\lambda_1 = 0$ and the corresponding eigenvector is constant.

Finally, the energy of a signal on graph can be localized without using any window, like in the case of the time-frequency energy distributions. An example of such a vertex-frequency distribution is introduced in [17] as

$$E(m, \lambda_k) = x(n)X(\lambda_k)u_k(n) = \sum_{m=1}^{N} x(n)x(m)u_k(m)u_k(n).$$

It satisfies the energy marginal properties as well [8], [9].

IV. Examples

The presented windowing methods are already illustrated on a simple graph with $N = 8$ vertices. The graph and windows centered at vertices 1, 2, 3 and 7 are presented in Fig. 1 for Method 1, Fig. 2 for Method 2 and Fig. 3 for Method 3. Vertices that belongs to the window support are shaded.

More complex graph with $N = 356$ vertices is considered next. Windows centered at vertices 51, 73, 171, and 279 are presented in Figs. 4, 5 and 6 for Methods 1, 2 and 3, respectively. Vertices outside window support are presented with small black dots, while large colored dots represent nonzero window values. It is evident that support for windows obtained by Method 1 is the whole graph, while the Methods 2 and 3 support is a subgraph defined as a neighborhood of the considered vertex.

The local vertex spectrum of a sample signal is presented in Fig 7. The signal on a graph is presented along with local vertex spectra obtained by using Method 1, 2 and 3. The sample signal is obtained as the sum of several delta functions and several eigenfunctions. It can be concluded that the presented methods result in very similar vertex spectrum representations, although Method 3 provides better concentration.

The concentration of the local vertex spectrum representation is measured using the normalized norm-one [16]

$$\mathcal{M} = \frac{\sum_{m=1}^{N}\sum_{k=1}^{N} |LVS(m, \lambda_k)|}{N \max_{m,k}\{|LVS(m, \lambda_k)|\}}. \tag{18}$$

This measure ranges from $1/N$ for best concentration (only one nonzero value in LVS) up to N for maximal spread (all LVS values are constant). For the considered case concentration measures obtained with Methods 1, 2 and 3 are

$$\mathcal{M}^{(1)} = 8.93$$
$$\mathcal{M}^{(2)} = 7.20$$
$$\mathcal{M}^{(3)} = 5.75,$$

meaning that the best concentration is achieved with Method 3. The window size can be optimized for each method using this concentration measure and the approach presented in [16].

V. Conclusion

Localizing a signal spectrum on a graph is considered in this paper. Three definitions of a localization windows set are considered, analyzed and applied to an example graph signal. It is shown that various windowing methods provide similar vertex spectral representations. The highest concentration in the vertex spectral domain is achieved by using the localization method based on the smallest path weight.

References

[1] S. Chen, R. Varma, A. Sandryhaila and J. Kovačević, "Discrete Signal Processing on Graphs: Sampling Theory," *IEEE. Trans. SP,* vol. 63, no. 24, pp. 6510–6523, 2015.

[2] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE. Trans. SP*, vol. 61, no. 7, pp. 1644—1656, 2013.

[3] A. Sandryhaila, J. M. F. Moura,"Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. SP*, no.12, pp.3042—3054, 2014.

[4] 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 SP Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[5] I. Jestrović, J. L. Coyle, and E. Sejdić, "Differences in brain networks during consecutive swallows detected using an optimized vertex–frequency algorithm," *Neuroscience*, vol. 344, pp. 113–123, 2017

[6] I. Jestrović, J. L. Coyle, and E. Sejdić, "A fast algorithm for vertex-frequency representations of signals on graphs," *Signal Processing*, vol. 131, pp. 483–491, 2017.

[7] A. Sandryhaila and J. M. F. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE SP Magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[8] L. Stanković, M. Daković, T. Thayaparan, *Time-Frequency Signal Analysis with Applications*, Artech House, Boston, 2013

[9] B. Boashash, ed, *Time-Frequency Signal Analysis and Processing, A Comprehensive Reference*, Academic Press, 2015.

[10] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-frequency analysis on graphs," *Applied and Computational Harmonic Analysis*, vol. 40, no. 2, pp. 260–291, 2016.

[11] D. I. Shuman, B. Ricaud, P. Vandergheynst, "A windowed graph Fourier transform," *SSP Workshop*, Aug. 2012, pp. 133-136.

[12] X. W. Zheng, Y. Y. Tang, J. T. Zhou, H. L. Yuan, Y.L. Wang, L. N. Yang, J.J. Pan, "Multi-windowed graph Fourier frames," *Machine Learning and Cybernetics (ICMLC), 2016 Int. Conf.*, Vol. 2, July, 2016, pp. 1042-1048.

[13] M. Tepper, S. Guillermo, "A short-graph fourier transform via personalized pagerank vectors," *Proc. IEEE ICASSP*, 2016.

[14] Lj. Stanković, M. Daković, E. Sejdić "Vertex-Frequency Analysis: A Way to Localize Graph Spectral Components," *IEEE Signal Processing Magazine*, vol. 34, July, 2017.

[15] M. Daković, L. Stanković, B. Lutovac, E. Sejdić, T. B. Šekara, "Resistive Circuits Analysis by Using Graph Spectral Decomposition," *6th Mediterranean Conference on Embedded Computing, MECO*, 2017

[16] L. Stankovic, "A measure of some time-frequency distributions concentration," *Signal Processing*, vol. 81, no. 3, pp.621-631, 2001

[17] L. Stankovic, E. Sejdic, M. Dakovic, "Vertex-Frequency Energy Distributions," *IEEE Signal Processing Letters*, submitted.

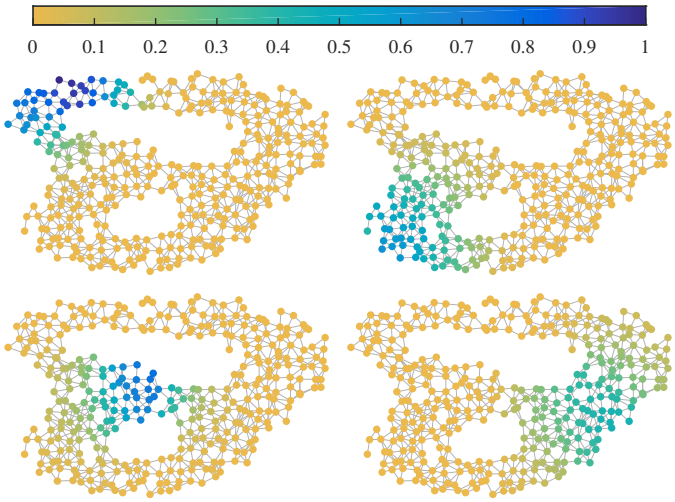

Fig. 4. Method 1: Windows centered at vertices 51, 73, 171 and 279. Window values at each vertex are presented with appropriate color.

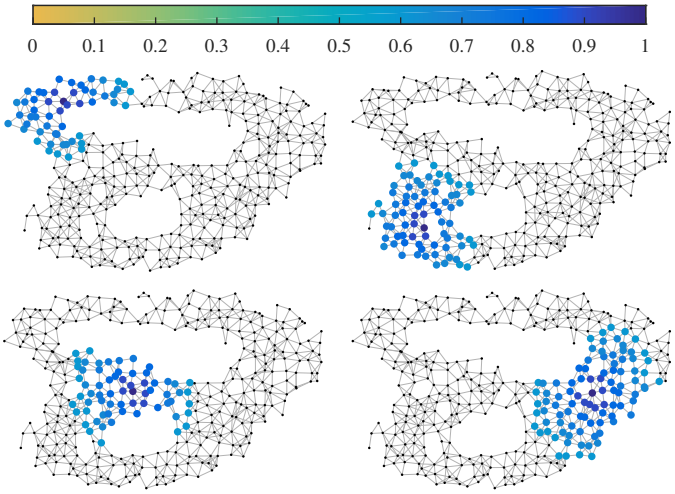

Fig. 5. Method 2: Windows centered at vertices 51, 73, 171 and 279. Window values at each vertex are presented with appropriate color.

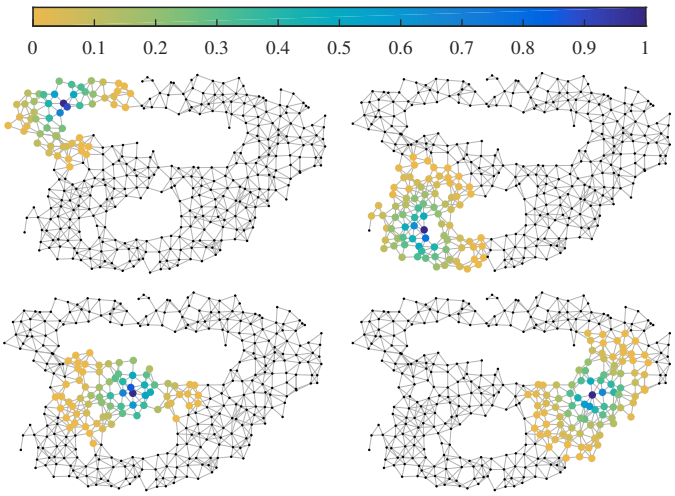

Fig. 6. Method 3: Windows centered at vertices 51, 73, 171 and 279. Window values at each vertex are presented with appropriate color.

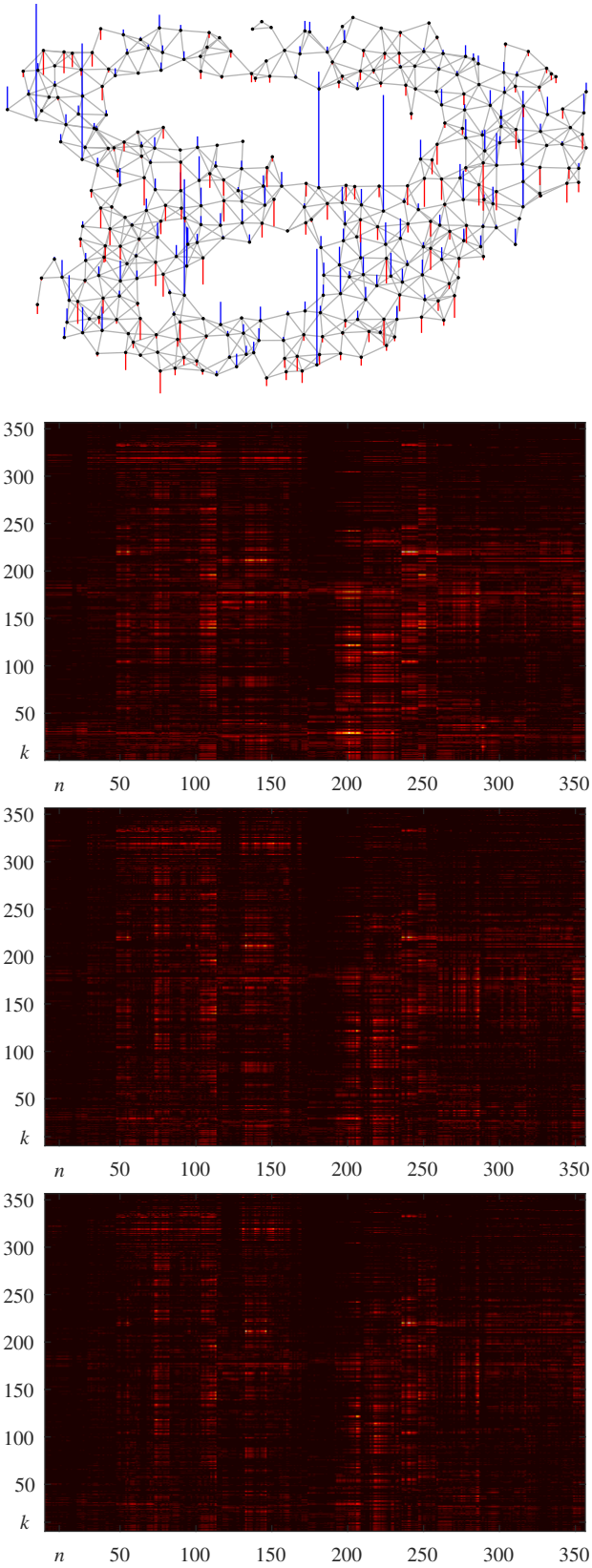

Fig. 7. Signal on graph (positive values are presented with blue and negative with red vertical bars) and local vertex spectrum representations obtained by windowing method 1, 2 and 3.