An Animated Visualization Method for Dynamic Scale-Free Networks*

Renato Fabbri[0000−0002−9699−629X] and Maria Cristina Ferreira de Oliveira[0000−0002−4729−5104]

University of São Paulo, São Carlos SP, Brazil
renato.fabbri@gmail.com, cristina@icmc.usp.br

Abstract. Most real-world networks are dynamic, i.e., they change over time as nodes and links are added or removed, and their representation may also involve variations in the associated data. There is a growing interest in modeling dynamic (or time-evolving, or longitudinal) networks and a number of procedures have been proposed to enable their visualization. This article presents a visualization method for dynamic scale-free networks. Named Versinus, the method consists of a specialized combination of layout and further visual cues useful in the observation of basic topological characteristics. The layout consists essentially in placing the most connected nodes (hubs and intermediary) along a sine curve, and the peripheral nodes along a separate segment, usually a line, thus its name, from the Latin *versus* (line) and *sinus* (sine). The visual cues yield evident both local and global key-features. A compliant method was not found in the scientific literature or among existing tools by the authors. Therefore, after a formal presentation of Versinus and its associated software implementations, a simple case study is described in order to provide insights into the potential exploitation of the method in analysing dynamic scale-free networks.

Keywords: Network visualization · Scale-free networks · Dynamic networks · Animated visualization · Complex networks · Data visualization.

1 Introduction

The visualization of dynamic networks poses challenges related to data size and complexity which have merited various contributions [1, 7, 9, 17]. In fact, the meaningful visualization of dynamic networks is a well-known problem in Information Visualization and dynamic network analysis. In this work, we present a peculiar method for dynamic network visualization which consists in distributing the nodes along a fixed layout and mapping the network evolution through changes in glyphs corresponding to nodes and links. The layout is arranged in three segments, each associated with one of the three sectors (or partitions) of a scale-free network, namely the hub, intermediary and peripheral nodes [5]. Local features of the network are mapped to the visual properties of the glyphs,

* Supported by FAPESP, project 2017/05838-3

e.g., glyph height, width, color and shape, and line (arrow) thickness and color. Moreover, complementary global and local information is rendered in order to enable the observation of basic aspects of the dynamic network. The method has been called Versinus since its early usages in 2013, because the layout settled, after attempts with distinct arrangements, in a sine period and a line, which in Latin is *sinus* (sine) and *versus* (line). No compliant method was found by the authors for visualizing dynamic scale-free networks through node-link diagrams. Furthermore, the visualization method was employed in scientific research [4, 5] and has received a number of software implementations.

The article is organized as follows. The next subsections address related work and nomenclature. Section 2 describes the method. Section 3 outlines the software implementations of Versinus and depicts the resulting animations and interfaces. Section 4 describes a simple case study in order to provide insights about the potentialities of the method. Section 5 is dedicated to hypotheses provided through the use of the method. Finally, Section 6 assembles concluding commentaries and further work.

1.1 Related work

Although animation for the purposes of data analysis is often criticized in the data visualization literature, advantages have also been reported [8, 13, 14, 21], e.g.: for eye catching visualizations and potential advantages yield by the fact that humans are used to interacting with moving objects. Dynamic network visualization has been tackled both through animated and static (often timeline-based) visualizations [9, 12, 19, 22], focusing on various global and local network features. The contribution reported in this document is possibly singular in that it consists basically of a layout which was designed for networks with scale-free characteristics, in association with some visual cues. It results from extensive attempts to reach a satisfactory visualization by means theoretical and intuitive strategies and methodical verification of the result by appreciation of the entailed animations.

1.2 Remarks on nomenclature

Dynamic networks are also called e.g. longitudinal networks, time-evolving networks, evolving networks, and time-varying networks [2, 3, 18, 20]. We adopted the term "dynamic network" herein because it is simple, semantically meaningful and has been widely adopted. Another major issue in current vocabulary arises when considered the hub, intermediary and peripheral sectors. In most cases, the sectors are also partitions (i.e. they are non-overlapping sets whose union is the complete superset), but it depends on the criteria for obtaining the sectors [5], thus the use of the more general term *sector*. Moreover, network science is highly multidisciplinary and context-dependent, and flourished from the vaguely-defined area of complex systems, favoring polysemy and synonymy. We strove to avoid such issues and to employ a consistent.

2 A description of the method

Versinus layout consists basically in placing the nodes in segments with respect to their overall topological features and then representing instantaneous topological features through glyph attributes. Additional, implementation-dependent, information is often provided. The achievement of the visualization may be thus split in three stages, described in the following subsections.

2.1 Node placement

First, a network is constructed with all the activity of interest, i.e. the transactions such as all the email messages considered, or all the links in a data file. The nodes are then ordered either by degree or strength (or in- or out-degree or strength), from greatest to smallest values. Also from such global network, the hubs, intermediary and peripheral sectors are obtained. Two methods have proven useful for obtaining the sectors: 1) assuming fixed fractions of hubs (e.g. 5%), of intermediary (e.g. 15%) and peripheral nodes (e.g. 80%); 2) comparison of the network against the Erdös-Rényi model, a non-trivial procedure, with relevant analytical results, detailed in [5].

With such structures at hand, the nodes are placed along three segments, as depicted in Figure 1. The hubs are placed along the first half of a sine period. The intermediary nodes are placed along the second half of the sine period. The peripheral nodes are placed along an independent line. These are the basic parameters:

- The amplitude α of the sine oscillation.
- The displacement Δ of the sine in the vertical axis.
- The horizontal margin μ, i.e. the distance from the left border to the first hub, assumed to be the same as the distance from the right border to last intermediary node.
- The positions (x_0, y_0) and (x_1, y_1), i.e. the endpoints of the segment for the peripheral sector.

Remarks: fine-tuning the size (number of nodes) of each sector may come handy due to the node density and activity in each segment. Also, the node placed at the middle of the sine period may be a hub or an intermediary node, and this choice often impacts the clarity of the final visualization more than anticipated by intuition, specially in small networks. Finally, avoiding clutter with the hubs motivated both the positioning of the peripheral sector in the upper-right corner and the placement of the most active peripheral nodes next to the intermediary sector, i.e. the ordering along the peripheral line.

2.2 Drawing nodes, links and auxiliary information

Let a *snapshot* S consist of a number σ time-contiguous transactions, e.g. 100 email messages, starting at an arbitrary transaction. The separation η between

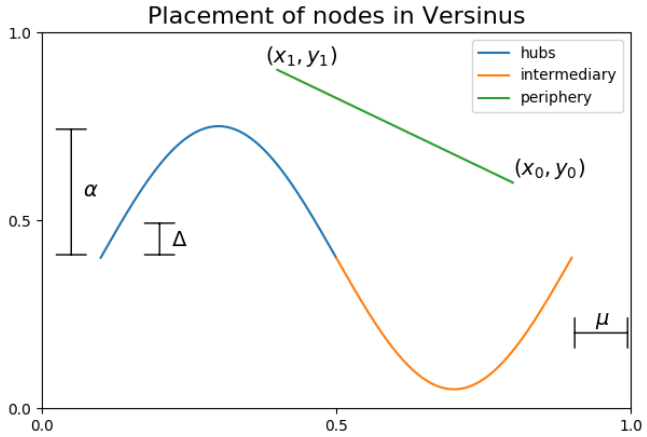


Fig. 1: The underlying layout of Versinus. Highly connected nodes are positioned from left to right on the sine segment (for the hubs and intermediary sectors), and from right to left on the upper line (for the periphery sector). The parameters are: amplitude α of the oscillation; vertical displacement Δ of the oscillation; horizontal margin μ; and line endpoints (x_0, y_0) and (x_1, y_1). Further information is provided in Section 2.1.

each snapshot is the number of transactions skipped from one snapshot to the next. The evolving network is thus conceptualized as a sliding window of σ transactions, which progresses by skipping η transactions at each step. To achieve the final animation, a transaction rate ω must also be defined. Thus, the following parameters are defined:

- The window size of σ captured at each snapshot, expressed in transactions.
- The step size η between each snapshot, expressed in transactions.
- The transaction rate ω, expressed in transactions per second.

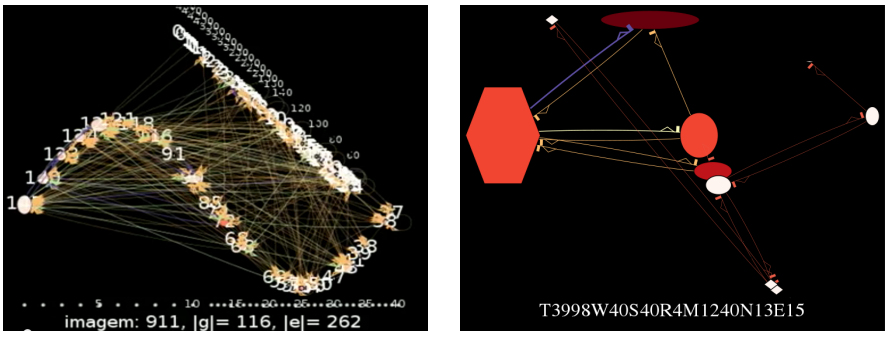
The sequence of snapshots may be expressed as $\{S_i^{\sigma,\eta}\}_0^L$, where L is the smallest integer such that $L \times \eta + \sigma > T$ and T is the total number of transactions. Notice that the final snapshot S_L may have less than σ transactions. Also, the rate ω is only applied when rendering the animation, e.g. if each snapshot S_i yields an image, the stop motion animation will use ω/η images per second to achieve the desired rate of ω transactions per second. In other words, the transactions in each snapshot are defined by σ and η and is independent from ω.

In each snapshot, the entailed network is analyzed in terms of the sectors (hubs, intermediary and periphery), in and out degree (or strength, if preferred) and clustering coefficient. Then the visual mapping of information is performed as follows:

- Nodes not incident in the snapshot are not shown. Glyphs associated with nodes incident in the snapshot are assigned a minimum width and height.

– The node sector in the snapshot is mapped to a glyph shape (e.g. circle, hexagon and diamond for hub, intermediary and peripheral nodes, respectively).
– Node width is incremented proportionally to its in-degree (or out-strength, if preferred). Node height is incremented proportionally to its out-degree (or out-strength if preferred).
– Node color is related to clustering coefficient, e.g. zero is mapped to white, one is pure red, and intermediary values are scaled accordingly.
– Link weight is mapped to line width and/or a color scale.

Finally, complementary information may be displayed as a persistent legend that reveals essential information such as σ, η, ω and T, and other information that assists the analyst in keeping track of essential features of the current snapshot, e.g. its number of nodes and edges. Also, node-specific information may be displayed by blinking corresponding cues over the nodes periodically, displaying them with a vertical displacement from the node, or by writing them to a specific widget when requested by the analyst on an interactive interface, i.e. on-demand. Figure 2 exemplifies two early Versinus settings.



(a) (b)

Fig. 2: Two frames of dynamic network animations achieved using the Versinus method. In (a) the individual degrees in the snapshot are blinking over the nodes, and the nodes are counted in auxiliary lines along the bottom and parallel to the periphery line. Current image/snapshot is 911, with 116 nodes and 262 links. In (b), a cleaner version is employed to display a snapshot in which the only hub is the overall hub, the second overall hub is a peripheral node, and most nodes have high clustering coefficient. The legend indicates that the total number of transactions is $T = 3,998$, the window size is $\sigma = 40$, the step size is $\eta = 40$, the transaction rate is $\omega = 4$, the current snapshot starts at transaction 1,240, and has 13 nodes and 15 links. The low-quality images are typical of the early implementations of Versinus, developed to assist the investigations of the researchers directly related to its development. Further information is provided in Section 2.

2.3 Observation of the evolving network

Within the settings described, one may notice a number of network characteristics: do nodes (or a specific node) change their sectors often? Is the overall intermediary sector in greater activity with the hubs or periphery? Does higher clustering occur most often in the less connected nodes? Are there very unbalanced nodes with respect to in and out degree (or strength)? Is the number of nodes and edges stable along the network evolution? In fact, Versinus was partly designed to inspect network attributes due to the remarkable stability of their overall characteristics reported in [5]. In this context, it is natural to ask if such stability is also noticeable in individual nodes or in the connectivity patterns. Although the overall characteristics may be regarded as surprisingly stable, the stability of network components is very network-dependent. E.g., as reported in [16], hubs exhibit intermittent activity in social networks, except for the smallest networks analyzed.

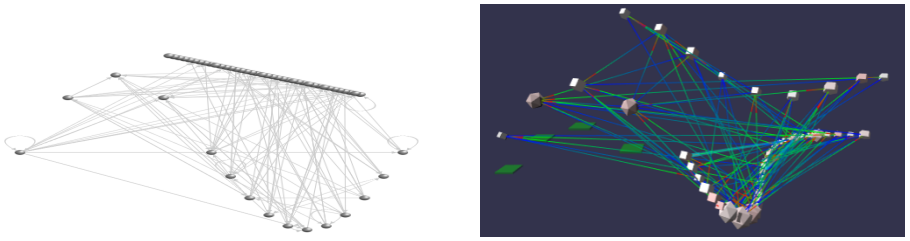
3 Software implementations

Currently, there are four software implementations of Versinus known to the authors. For organization and brevity, these are all linked in[1]. Two of them are written in Python: one is a script that renders animations as in the Figure 2 (a), with more cues of local structure. The other is a library that renders animations as in the Figure 2 (b), simpler and with more cues of the overall structure, and is capable of rendering sonifications of the dynamic networks, yielding "audiovisualizations", i.e. synced sonic and visual mappings of the dynamic networks, as described in Section 3.1. The other two implementations are achieved through JavaScript and WebGL. One is, by default, for static (non-dynamic) networks, implemented as part of the ccNetViz library, that aims at being as computationally inexpensive as possible [10]. The other is an experimental interactive visualization application in which Versinus is combined with other analytical techniques, such as principal component analysis, overview first - focus on demand, and simultaneous highlighting of corresponding structures across images as required by users. These last two implementations are illustrated in Figure 3.

3.1 Network sonification

The implementation illustrated in Figure 2 (b) includes a sonification routine for the evolving network. Results are of limited effectiveness in the sense that the sonification does not enable users to grasp network features with ease, specially if they are not musically trained. Nevertheless, it made the (audio)visualization more engaging, as it yields exotic media, and it is the first time the authors appreciated a sonification of an evolving network (maybe even of any network). It is possibly a unique multimedia representation of networks. Accordingly, we

[1] Central repository for Versinus resources such as implementations, public instances of interactive interfaces, and videos: https://github.com/ttm/versinus



(a) Versinus in ccNetViz

(b) Experimental interactive visualization

Fig. 3: Two implementations of Versinus in JavaScript using WebGL. (a) illustrates the implementation that is available within ccNetViz, a library purposed to being computationally inexpensive. This implementation handles static networks by default. (b) is an image achieved using an experimental interactive visualization application. This implementation renders the animations in real time, with multiple controls for inspecting the network structure and controlling the animation, in association with other analysis techniques. The sine oscillates both in the vertical and depth axes, as does the periphery segment.

understand suitable to provide a short description of the sonification as it was left implemented, i.e. in its final version, even though a full understanding of the summary requires knowledge of musical theory [6]. Four sonic/musical voices are used, two of them are noises and the other two have defined pitch. The lower noise has central frequency proportional to first hub connectivity, which is the node positioned at the extreme left on the layout, on the starting point of the sine, and its frequency is updated every two beats. The higher noise has central frequency proportional to the second hub, the node next to the first hub, and its frequency is updated at each beat. The third hub connectivity is proportional to the lower note, which is incident once a beat, and has its note value updated at every beat. The fourth hub connectivity is proportional to the higher note, which is incident twice every half-beat, is updated at every note, and is only heard on the second half-beat of every tempo. Figure 4 illustrates such disposition of sonic lines (or musical voices).

3.2 Animations produced and available

A number of animations were obtained through the routines described. They encompass different networks, settings (α, σ, η, etc), and different flavors of Versinus. For organization and brevity, they are available at [2].

[2] Central repository for Versinus resources such as implementations, public instances of interactive interfaces, and videos: https://github.com/ttm/versinus

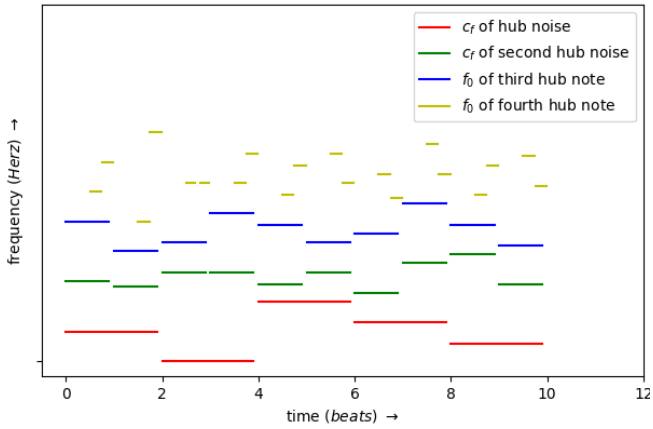


Fig. 4: Visual representation of the sonification implemented for the dynamic networks. c_f and f_0 stands for *center frequency* and *fundamental frequency*, respectively. Further information is provided in Section 3.1.

4 A simple case study

This section exemplifies the employment of the Versinus network visualization method within a more elaborate framework through a very simple case study. Let us consider an interactive visual interface, with which the user may parametrize Versinus and then navigate through the entailed dynamic network. A most fundamental Versinus parametrization is performed before the visual mapping of the network, and is illustrated in Figure 5. Initial visualization is illustrated in Figure 6. Figures 7 and 8 exemplifies Versinus compliant interactive data visualization procedures: the observation of the nodes in each sector, and characteristics of individual nodes and of snapshots. The interactive interface is named $ESFNetVis$ (after "Evolutive Scale Free Network Visualization"). Directions for accessing the code and operating instances of $ESFNetVis$ are provided in[3].

One basic feature of the method is the constant representation of nodes in each of the hubs, intermediary and peripheral sectors. Nodes are positioned according to such classification regarding all the activity of interest, and in each snapshot visual cues map the classification within the snapshot. Following the analytic methods used in [5], it may be crucial to know how the relative sizes of the sectors vary along time, i.e. along the sequence of snapshots, and how often nodes migrate between sectors.

[3] Central repository for Versinus resources such as implementations, public instances of interactive interfaces, and videos: https://github.com/ttm/versinus

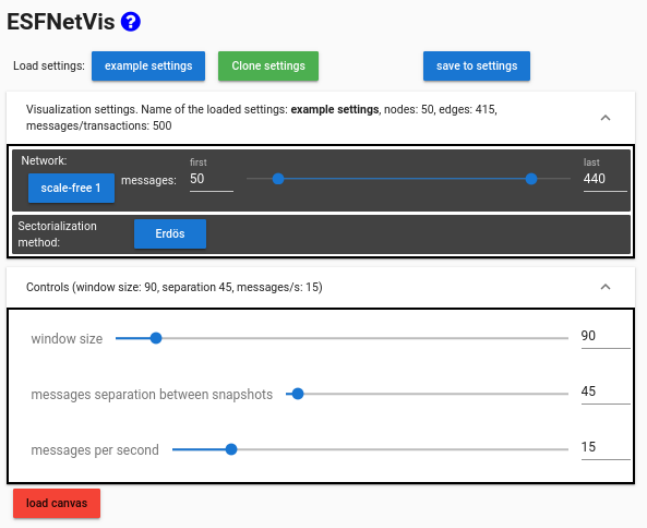


Fig. 5: Visual elements for the settings performed before the visual mapping of the dynamic network. The buttons at the top are for keeping, modifying and accessing settings. The widgets in the gray rectangles enables the selection of the network, the first and last messages (transactions) to be considered, and the sectorialization method. The widgets in the white rectangle are σ, η, and ω. These settings are compliant with the description available in Section 2. The "Erdös" setting stands for the achievement of the network sectors by comparing the network against an Erdös-Rényi model, mentioned in Section 2.1. When the red button is clicked, the dynamic network is mapped to the canvas and additional elements, as described in Figure 6.

5 Hypotheses gathered

This section describes hypotheses collected by the authors, mostly about dynamic network layouts, through the development and usage of Versinus. Most of these hypotheses should be further verified in subsequent work and using suitable validation methods.

The visualization of network structures is favored by the placement of nodes in an oscillatory pattern, as it tends to decrease the superposition of nodes and links. In fact, this principle was implemented in some of the layouts available within ccNetViz, and it may be useful for other well-known layouts, such as for the Hive Plot [11]. Using a single period for the sine is an arbitrary choice, established because it entailed good visualizations for the networks and tasks at hand during the design of Versinus. Nonetheless, it is should be allowed to vary in further implementations. Another important insight is the pertinence of fixed positions for the observation of dynamic networks. Not only this has the potential to reduce the hairball effect [15], but it is most often very difficult to

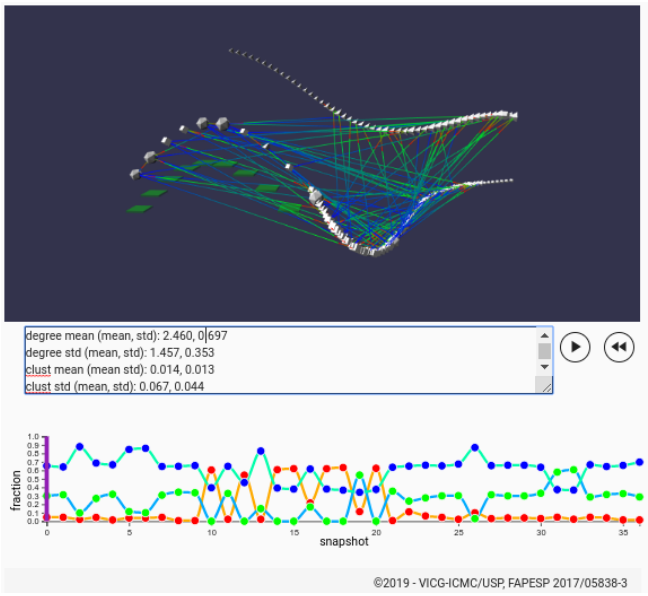


Fig. 6: The initial visualization of the dynamic network performed using $ESFNetVis$. In the canvas, the network is mapped using a 3D extension of Versinus, in which both segments oscillates vertically and in depth. The red portions of the links denotes the direction of the link (often achieved using arrow heads). The text area starts with overall information of the collection of snapshots (e.g. each snapshot yields a mean degree, the set of snapshots yields a mean of such mean degree and a standard deviation of the mean degree). The timelines hold the fractions of nodes in each sector with red, green and blue for hub, intermediary and peripheral nodes. The red bar is the playback cue, i.e. the snapshot being visualized. The analyst may play and pause with the play button, and rewind with the rewind button. It is evident from the timelines that the expected fraction of nodes in each sector is not preserved throughout the network evolution. This entails that in some snapshots the scale-free distribution of connectivity is degraded.

keep track of individual nodes and structures when the nodes are allowed to move in the images while the network evolves, e.g. in using a force-based layout which is constantly updated. The placement of nodes with respect to their topological features (nodes are ordered by connectivity, segments are related to network sectors) revealed very useful for understanding network features, such as stability and connectivity-dependent behaviors. Visual cues such as glyph width, height, shape and color were found informative for grasping information on individual nodes, but often result in a cognitive overload when attempting to observe global features, which motivates simplified settings. The sonification of the evolving network added to the exotic flavor of Versinus and made it

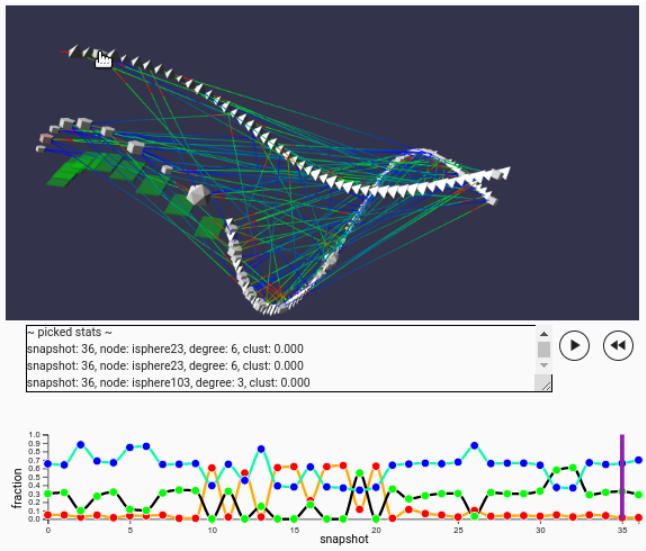


Fig. 7: The layout is slightly rotated in order to focus on the peripheral nodes segment. This image depicts one of the interaction possibilities among the canvas and the timelines: when the mouse is over a node, the line of the sector of the node in the snapshot is made black. The overall peripheral node is an intermediary node in this snapshot. (Notice the pointer in "button-mode", i.e. visualized as a hand.) Also, the text area in this image shows information about specific nodes, provided on-demand to the user, which hits the key "i" when the mouse is over a node for the interface to provide the information.

more attractive. However, it seems that more informative sonifications should require dedicated research, training of already musically competent users, and very modest goals. Furthermore, reasonable parametrization (e.g. through α, Δ, μ, (x_0, y_0), (x_1, y_1), σ, η and ω) is highly dependent on the network characteristics and user intended tasks, and calls for interactive interfaces and automated recommendation of initial values.

6 Conclusions and further work

This document presents a description of the Versinus visualization method, with its fundamental elements, usage potentials, and software implementations. The validity of Versinus is implied both by its employment in published research [5, 4], and the insights it yields on individual networks and network layouts in general. Further developments, with focus on interactivity and Web technologies, are currently being refined through research and software development.

Next steps include automated tuning of Versinus parameters, implementation of cues for relating language incident in the networks to their topology,

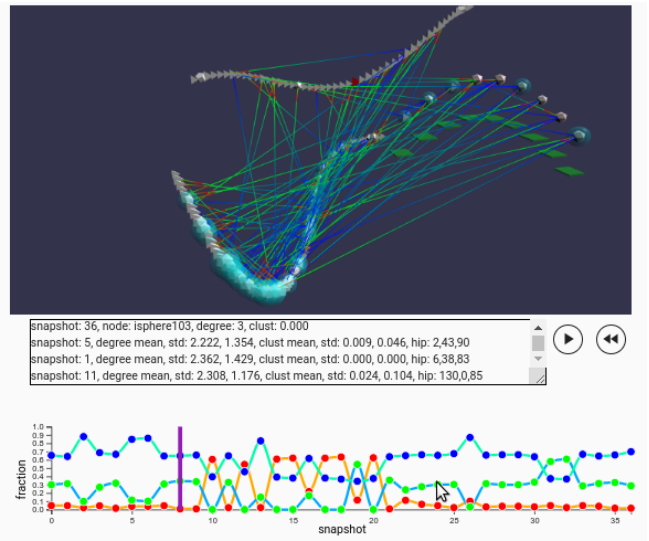


Fig. 8: The layout is rotated in order to focus on the intermediary segment of the sinusoid. The text area shows information about specific snapshots, given on-demand to the analyst, which hits "S" while the snapshot is being visualized. The nodes in the intermediary sectors, visualized on the canvas, are being highlighted by the user: notice the mouse pointer which is over the intermediary sector timeline and the transparent blue spheres around the nodes.

and further visually representing topological features (e.g. the fraction of the interaction of the intermediary sector that is shared with the hubs and with the periphery). The experimentation of Versinus features within other layouts is a potentially valuable contribution, e.g. wiggling the segments of a Hive Plot [11]. We should implement further mechanisms for the detection of structural changes within the network evolution, such as persistent homology [9]. Most importantly, we aim at completing a Visual Analytics utility to support using Versinus in an interactive environment for exploratory data analysis, and verifying the claims described in Section 5 through formal validation techniques. Finally, we plan to further analyze data sets of evolving networks [5] using Versinus.

Acknowledgements The authors acknowledge the financial support of the São Paulo State Research Foundation (FAPESP grant 2017/05838-3). The views expressed do not reflect the official policy or position of FAPESP.

References

1. Albert, R., Barabási, A.L.: Topology of evolving networks: local events and universality. Physical review letters **85**(24), 5234 (2000)

2. Bianconi, G., Barabási, A.L.: Competition and multiscaling in evolving networks. EPL (Europhysics Letters) **54**(4), 436 (2001)
3. Bogdanov, P., Mongiovì, M., Singh, A.K.: Mining heavy subgraphs in time-evolving networks. In: 2011 IEEE 11th International Conference on Data Mining. pp. 81–90. IEEE (2011)
4. Fabbri, R.: Topological stability and textual differentiation in human interaction networks: statistical analysis, visualization and linked data. Ph.D. thesis, Universidade de São Paulo (2017)
5. Fabbri, R., Fabbri, R., Antunes, D.C., Pisani, M.M., de Oliveira Junior, O.N.: Temporal stability in human interaction networks. Physica A: Statistical Mechanics and its Applications **486**, 92–105 (2017)
6. Fabbri, R., Junior, V.V.d.S., Pessotti, A.C.S., Corrêa, D.C., Oliveira Jr, O.N.: Musical elements in the discrete-time representation of sound. arXiv preprint arXiv:1412.6853 (2014)
7. Fenu, C., Higham, D.J.: Block matrix formulations for evolving networks. SIAM Journal on Matrix Analysis and Applications **38**(2), 343–360 (2017)
8. Fisher, D.: Animation for visualization: opportunities and drawbacks. Ch **19**, 329–352 (2010)
9. Hajij, M., Wang, B., Scheidegger, C., Rosen, P.: Visual detection of structural changes in time-varying graphs using persistent homology. In: 2018 IEEE Pacific Visualization Symposium (PacificVis). pp. 125–134. IEEE (2018)
10. Helikar, T., Kowal, B., McClenathan, S., Bruckner, M., Rowley, T., Madrahimov, A., Wicks, B., Shrestha, M., Limbu, K., Rogers, J.A.: The cell collective: toward an open and collaborative approach to systems biology. BMC systems biology **6**(1), 96 (2012)
11. Krzywinski, M., Birol, I., Jones, S.J., Marra, M.A.: Hive plots—rational approach to visualizing networks. Briefings in bioinformatics **13**(5), 627–644 (2011)
12. Ma, C., Kenyon, R.V., Forbes, A.G., Berger-Wolf, T., Slater, B.J., Llano, D.A.: Visualizing dynamic brain networks using an animated dual-representation. In: Proceedings of the Eurographics conference on visualization (EuroVis). pp. 73–77 (2015)
13. Munzner, T.: Visualization analysis and design. AK Peters/CRC Press (2014)
14. Nakakoji, K., Takashima, A., Yamamoto, Y.: Cognitive effects of animated visualization in exploratory visual data analysis. In: Proceedings Fifth International Conference on Information Visualisation. pp. 77–84. IEEE (2001)
15. Nocaj, A., Ortmann, M., Brandes, U.: Untangling hairballs. In: International Symposium on Graph Drawing. pp. 101–112. Springer (2014)
16. Palla, G., Barabási, A.L., Vicsek, T.: Quantifying social group evolution. Nature **446**(7136), 664 (2007)
17. Peel, L., Clauset, A.: Detecting change points in the large-scale structure of evolving networks. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
18. Perra, N., Gonçalves, B., Pastor-Satorras, R., Vespignani, A.: Activity driven modeling of time varying networks. Scientific reports **2**, 469 (2012)
19. Schneider, B., Acevedo, C., Buchmüller, J., Fischer, F., Keim, D.A.: Visual analytics for inspecting the evolution of a graph over time: pattern discovery in a communication network. In: 2015 IEEE Conference on Visual Analytics Science and Technology (VAST). pp. 169–170. IEEE (2015)
20. Vu, D.Q., Hunter, D., Smyth, P., Asuncion, A.U.: Continuous-time regression models for longitudinal networks. In: Advances in neural information processing systems. pp. 2492–2500 (2011)

21. Ware, C.: Information visualization: perception for design. Elsevier (2012)
22. Wu, Y., Pitipornvivat, N., Zhao, J., Yang, S., Huang, G., Qu, H.: egoslider: Visual analysis of egocentric network evolution. IEEE transactions on visualization and computer graphics **22**(1), 260–269 (2016)