Versinus: 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 visualize them. This article presents a visualization method for dynamic scale-free networks. Compliant methods were not found in other tools by the authors, which motivated this description and the development of a dedicated interface. Named Versinus, the visualization is specially useful in the observation of (in)stability of basic topological characteristics, and 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 Latin *versus* (line) and *sinus* (sine). The Versinus visualization can provide insight into network properties and is useful for observing its behavior, as described in this paper, after a formal presentation of the method and its associated software implementations.

Keywords: Network visualization 12
· Dynamic networks · Longitudinal networks · Time-evolving networks · Evolving 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, 9, 7, 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 an approach for dynamic network visualization that consists in distributing the nodes along a fixed layout arranged in three segments associated with the three sectors (or partitions) of a scale-free network, namely for the hub, intermediary and peripheral nodes [6]. The network dynamic features are expressed by means of changes in visual properties of the glyphs used to depict nodes and links, e.g., glyph height, width, color and shape, and line (arrow) disposition, thickness and

* Supported by FAPESP, project 2017/05838-3

color. Moreover, complementary global and local information is rendered in order to enable an acoustic depiction of features of interest. The method has been called Versinus since its early usages in 2013, because the layout stabilized, after attempts with distinct arrangements, in a sine period and a line, which in Latin is *sinus* (sine) and *versus* (line). There are four software implementations of Versinus, two of them written in Python scripts and packages that yield animations through stop-motion. The other two are written in JavaScript, one focused on exploitation of static networks, the other is an experimental Visual Analytics interface, which extends the visual metaphor to the 3D space and provides additional analysis techniques such as principal component analysis (PCA) and resources for structure navigation[1]. The writing of this document was motivated by the fact that no method was found by the authors that aims at visualizing dynamic scale-free networks through node-link diagrams. Furthermore, the visualization method was employed in scientific research [5, 6], and thus a careful mathematical and technological description of the procedures involved in applying Versinus to real networks is appropriate.

The article is organized as follows. The next subsections address related work and issues of nomenclature. Section 2 describes the method in detail and sufficient formalism. Section 3 outlines the software implementations of Versinus and depicts the resulting animations. Section 4 is dedicated to revealing further insights obtained through the use of the method. Finally, Section 5 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.: yielding visualizations more eye catching and taking advantage of 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 singular in that it consists basically of a layout which is fit for networks with scale-free characteristics, in association with some visualization cues which are implementation dependent. 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. The procedures and conceptualizations involved in achieving the final visualization were also considered worth documenting, as they were not found elsewhere in the literature, and by virtue of the simple and genuine design. Its scientific relevance on a practical application has been demonstrated before [6].

[1] An instance of the interface is available at: http://rfabbri.vicg.icmc.usp.br:3000/evolution

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 chose to adopt the term "dynamic network" herein because it is simple, semantically meaningful and has been widely adopted. Another major issue in current vocabulary is in the phrase "sector of a network" (e.g. of the hubs). 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 [6], thus the use of the more general term *sector*. Moreover, network science is highly multidisciplinary and context-dependent, and flourished from the ill-defined area of complex systems, favoring polysemy and synonymy. We strove to avoid such issues and keep a consistent usage of nomenclature.

2 A description of the method

Versinus generates a layout through placing the nodes in segments with respect to their overall topological features and then representing instantaneous topological features though glyph and arrow attributes. 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 been proven useful for obtaining sectors: 1) assuming fixed fractions of hubs (e.g. 5%), of intermediary (e.g. 15%) and peripheral nodes (e.g. 80%); 2) by comparison of the network against the Erdös-Rényi model, a non-trivial procedure, with relevant analytical results, detailed elsewhere [6].

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. The placement follows 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 hubs, assumed 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.

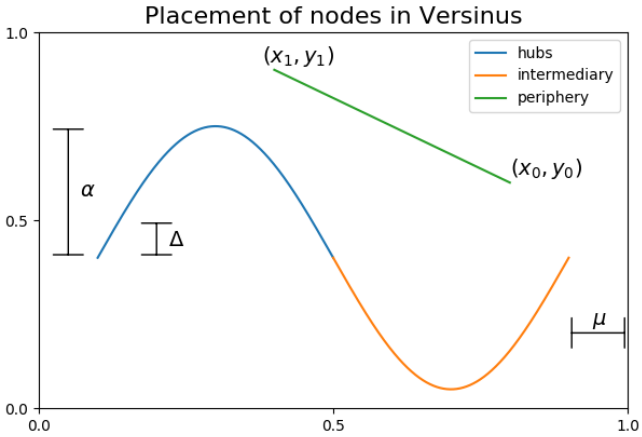


Fig. 1: The underlying layout of Versinus. Highly connected nodes are positioned from left to right on the sine segment (for the hubs and intermediary sectors), and from right to left on the upper line (for the periphery sector). The parameters are: amplitude α of the oscillation; vertical displacement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

Some crafty 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. 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

Additional concepts and quantities need to be defined for visualizing the evolving network as a data-driven animation. A *snapshot* S consists of a number σ time-contiguous transactions, e.g. 100 email messages, starting at an arbitrary message. The separation η between each snapshot is the number of transactions skipped from one snapshot to the next. In summary, the evolving network is visualized through a sliding window of σ transactions, updated at every η transactions. To achieve the final animation, a transaction rate ω must also be defined. Thus, the following parameters are specified:

- the window 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.

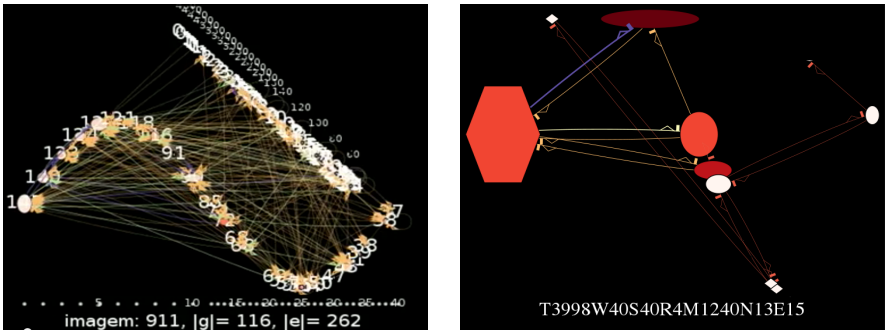
In each snapshot, the entailed network is analyzed in terms of the sectors (hubs, intermediary and periphery), in and out degree (or strength, if preferred) and clustering coefficient. Then the visual mapping of information is performed as follows:

- 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 keeps track 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 of the early flavors of Versinus settings.

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 really 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 [6]. In this context, it is natural to ask if such stability is also noticeable in individual nodes or in connectivity patterns. Although the overall characteristics may be regarded as surprisingly stable, the stability of network components is very network-dependent. E.g., as suggested by [16], all hubs were found to have intermittent activity in social networks, except for the smallest networks analyzed.



(a) (b)

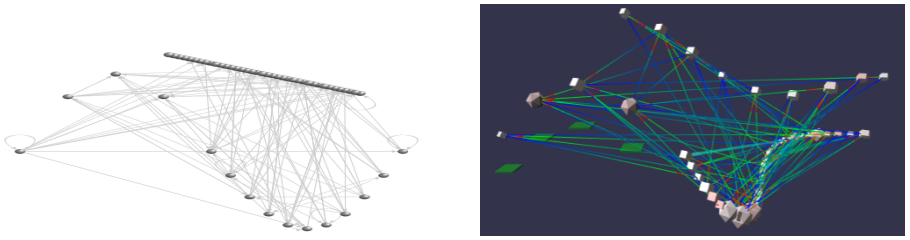
Fig. 2: Two frames of dynamic network animations achieved using the Versinus method. Both are achieved through dedicated Python scripts and package modules. In (a) the individual degrees in the snapshot are blinking over the nodes, and the nodes are counted in auxiliary lines along the bottom and paralel to the periphery line. Current image/snapshot is 911 and has 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 $\sigma = 40$, 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 given in Section 2.

3 Software implementations

Currently, there are four software implementations of Versinus known to the authors. For organization and brevity, these are all linked in[2]. 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. The other two implementations are in JavaScript and use 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 3D Visual Analytics 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 struc-

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

tures across images as required by users. These last two implementations are illustrated in Figure 3.



(a) Versinus in ccNetViz (b) Experimental Visual Analytics

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 Visual Analytics application. It 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. Further information is given in Section 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), and is thus probably a unique multimedia representation of networks. Accordingly, we 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. 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 each two tempos. The higher noise has central frequency proportional to the second hub, the node next to the first hub, and its frequency is also updated every two tempos. The third hub connectivity is proportional to the lower note, which is incident once a beat, and has its note value updated accordingly. The fourth hub connectivity is proportional to the higher note, which is incident twice every half-beat, is updated every note, and

is only heard on the second half-beat of every tempo. Figure **??** illustrates such disposition of sonic lines (or musical voices).

3.2 Animations produced and available

Many animations were obtained through the routines described. These encompassed different networks and settings (α, σ, η, etc), and different flavors of Versinus. Some of these animations are available in six playlists of Youtube videos. For organization and brevity, they are available at [4] together with other Versinus-related resources.

4 Insights gathered

This section describes insights gathered by the authors, and should be further verified in subsequent work and using suitable validation methods. The legitimacy of the exposition herein given is based on the reasonable aspect of the assertions, and on the facts that the authors developed Versinus, used the method extensively, and the scientific contribution which it made possible [6].

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 in other well-known layouts, such as 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 allowed to vary in more recent implementations in JavaScript listed in [3]: the sine in which the hub and intermediary nodes are positioned may have as many periods as parametrized by the analyst, and the same is feature is available for the segment for the peripheral nodes. Another important insight is the pertinence of fixed positions for the observation of evolving networks. Not only this has the potential to reduce the hairball effect [15], but it is most often very difficult to keep track of individual nodes and structures when the nodes are allowed to move in the images while the network evolves, as in force-based layouts. 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 results 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 more attractive. However, it seems that more informative sonifications should require dedicated research, training of already musically competent users,

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

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.

5 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 [6, 5], 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, and further visually representing topological features (e.g. the fraction of interaction of the intermediary sector 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 using the persistent homology [9]. Most importantly, we aim at completing a Visual Analytics utility to support using Versinus in an interactive environment for enhanced exploratory data analysis, and verifying the claims described in Section 4 through formal validation techniques. Finally, we plan to further analyze data sets of evolving networks [6] using Versinus.

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.: The git repository for indicating versinus resources (2019), https://github.com/ttm/versinus
5. 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)
6. 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)
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)