Versinus: an Animated Visualization Method for Evolving Networks*

Renato Fabbri$^{[0000-0002-9699-629X]}$ and Maria Cristina Ferreira de Oliveira$^{[0000-0002-4729-5104]}$

University of São Paulo, São Carlos SP, BR
renato.fabbri@gmail.com, cristina@icmc.usp.br
http://conteudo.icmc.usp.br/pessoas/cristina/

Abstract. Most real-world networks change over time with new nodes and links or their removal, and their representation may involve metadata variation. Increasing interest in modeling evolving networks has been reported and a number of procedures have been proposed to visualize them. This article presents a novel visualization method for evolving (or longitudinal, or time-evolving) networks, specially useful in the observation of (in)stability of basic topological characteristics. The method has received a few software implementations and has been useful for research and publication, which motivated the writing of this present document. Named Versinus, the visualization 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 method has provided insights into network properties and for network visualizations, which are described here after a formal presentation of the visualization and yielded software.

Keywords: Network visualization · Evolving networks · Animated visualization · Complex networks · Data visualization.

1 Introduction

The visualization of evolving networks poses challenges related to data size and complexity which have merited various contributions [1, 5, 14]. In fact, the meaningful visualization of time-evolving networks is a well-known problem in information visualization (*infovis*) and dynamic network analysis (*DNA*). In this work, we present a novel approach in DNA infovis that consists in distributing the nodes along a fixed layout that consists in three segments for three sectors (or partitions) of a scale-free network, namely for the hub, intermediary and peripheral nodes [4]. The development of the network is expressed by changes in node height, width, color, format, and by changes in link disposition, thickness and color. Also, auxiliary global and local information are be displayed in order to enable a sound analysis with the features of interest. The method has

* Supported by FAPESP, project 2017/05838-3

been called Versinus since its early usages in 2013, because the layout stabilized, after trying many figures, in a sine period and a line, which in Latin is *sinus* (sine) and *versus* (line). Versinus has received four software implementations, two of them written in Python scripts and packages that yield animations through stop-motion. The other two are written in JavaScript, one of them is an exploitation for static networks, the other is an ongoing Visual Analytics interface which extends the description in this document for 3D with additional analysis techniques such as principal component analysis (PCA) and structure navigation. The writing of this document was motivated by the usefulness of the method in scientific research [3, 4] and the benefit of a careful mathematical and technological description of the procedures involved in applying the method to real networks.

The article is organized as follows. Next subsections hold remarks about related work and vocabulary. Section 2 describes the method in detail and sufficient formalism. Section 3 depicts the software implementations of Versinus and resulted animations. Section 4 is dedicated to revealing further insights obtained through the use of the method. Finally, Section 5 assemble concluding commentaries and further work.

1.1 Related work

Although animation is often criticized in the data visualization canonical literature, advantages have also been reported [6, 10, 11, 16] e.g. making visuallizations more eye catching and due to the fact that we are accustumed to moving through the world and to moving objects. Evolving network visualization has been tackled both through animated and static (often timeline-based) visualizations [17, 9, 15], focusing on various global and local network features. The contribution reported in this document is singular in consisting basically of a network layout which is fit for scale-free characteristics, together with some visualization cues dependent on the implementation, and in being the result of extensive attempts to reach a satisfactory visualization by theoretical and intuitive strategies and methodical verification of the result by appreciation of the entailed animations. The procedures and conceptualizations involved in the achieving the final visualization were also considered worth documenting as they were not found elsewhere in the literature, were designed simple and genuine, and found of scientific relevance [4].

1.2 Nomenclature remarks

Evolving networks are also called e.g. logitudinal networks, time-evolving networks, dynamic networks. We chose the term "evolving network" herein because it is simple, is semantically reasonable, and has been very much in use. The other main 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-overlaping sets whose union is the complete superset), but it depends of the criteria for obtaining the sectors [4], thus the use of the more general vocable *sector*. Moreover, network science is very multidisciplinary and context-dependent, and flourished from the ill-defined area of complex systems, favoring polysemy and synonymy. We strove to avoid such issues because they are usually not convenient in scientific literature.

2 A description of the method

Versinus is essentially obtained through placing the nodes in segments with respect to their overall topological features, and then representing instantaneous topological features though gliph and arrow attributes. The visualization may be divided thus 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. all 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 such 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, carefully described in [4].

With such structures at hand, the nodes are placed along three segments. 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 indepentent line. Such placement is depicted in Figure 1 with 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.

Some crafty notes: fine-tunning of the size of each sector (number of nodes) may come handy due to the density of nodes in each segment and due to their activity. Also, the node positioned at the middle of the sine period may be a hub or an intermediary node, and this choice often impacts final visualization clarity more than predicted 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.

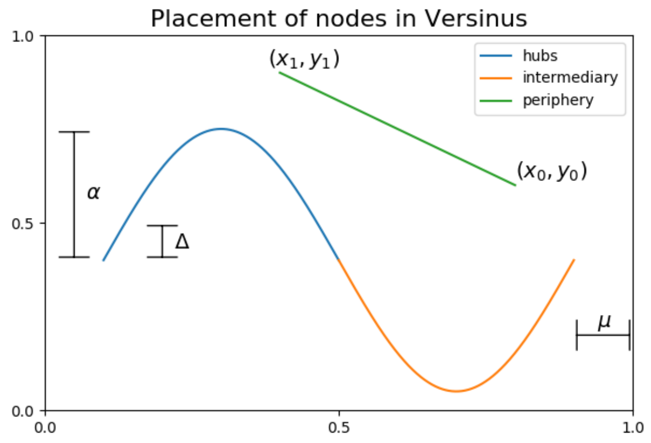


Fig. 1: The underlying layout of Versinus. Most connected nodes are positioned from left to right on the sine segment (for the hubs and intermediary sectors), and from right to left on the upper line (for the periphery sector). The parameters are: amplitude α of the oscillation; vertical displacement 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.

2.2 Drawing nodes, links and auxiliary information

For visualizing the evolving network, further concepts and quantities need to be defined. A *snapshot* S consists of a number σ time-contiguous transactions, e.g. of 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 each η transactions. To achieve the final animation, one need also to define a transaction rate ω. Thus, these parameters are specified:

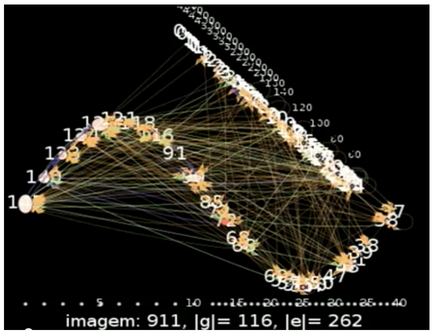
- the window σ captured at each snapshot, expressed in transactions.
- The step size η between each snapshot, expressed in transactions.
- The transaction rate ω, expressed in transactions per second.

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 last snapshot S_L will not necessarily have σ transactions, but may have less than σ. 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.

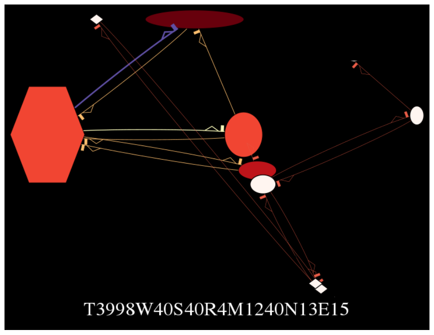
In each snapshot, the entailed network is analysed in terms of the sectors (hubs, intermediary and periphery), in and out degree (or strength if proffered) and clustering coefficient. Then the visual mapping of information is performed in the following way:

– nodes not incident in the snapshot are not shown. Nodes incident in the snapshot is given a minimum width and height.
– Node sector in the snapshot is mapped to node shape (e.g. hexagon, circle and diamond for hub, intermediary and peripheral nodes).
– Node width is incremented proportinally to its in-degree (or out-strength if preferred). Node height is incremented proportionally to its out-degree (or out-strength if preffered).
– 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 link width and/or a color scale.

Finally, auxiliary information may be given by a persistent legend that displays essential information such as σ, η, ω and T, and keeps track of current snapshot along e.g. the number of nodes and edges of the snapshot. Also, node-specific information may be displayed by blinking corresponding cues over the nodes periodically or displaying them with a vertical displacement. Figure 2 examplifies two flavours of Versinus settings.



(a) Versinus flavour A



(b) Versinus flavour B

Fig. 2: Two frames of animations achieved using the Versinus method. In (a) the individual degrees in the snapshot is blinking over the nodes, and the nodes are counted in auxiliary lines along the bottom and the near the periphery line. Current image/snapshot is 911 and the snapshot has 116 nodes and 262 links. In (b), a cleaner version of Versinus is used 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 coefficients. The legend indicates that the total number of transactions is $T = 3998$, the window size $\sigma = 40$, the step size is $\eta = 40$, the transaction rate is $\omega = 4$, current snapshot starts at transaction 1240, and the snapshot has 13 nodes and 15 links. Further information is given in Section 2.

2.3 Observation of the evolving network

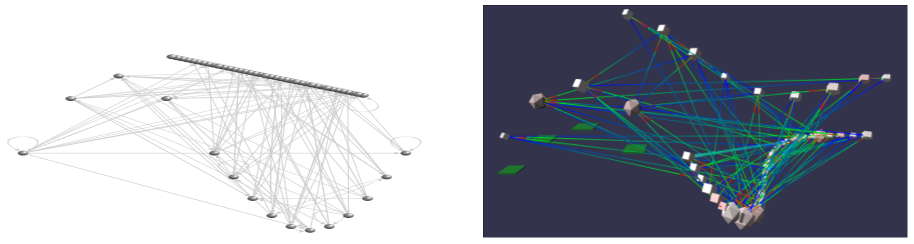
Within the settings described, one may notice a number of network characteristics: do nodes (or a specific node) change their sectors often? Is the overall intermediary sector in greater activity with the hubs or periphery? Do higher clustering really occur most often in the less connected nodes? Are there very unbalanced nodes with respect to in and out degree (or strength)? Are 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 stablity of their overall characteristics reported in [4]. In this context, it is natural to ask if such stability is also noticeable in individual nodes, and, as suggested by [13], all hubs were found to have intermittent activity in social networks, except for the smallest networks analysed.

3 Software implementations

There are four software implementations of Versinus. For organization and brevity, these are all linked in this repository: [2], together with this document. 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 (a), simpler and with more cues of the overall structure. The other two implementations are in JavaScript and uses WebGL. One is for static (non-evolving) networks, implemented as part of the ccNetViz library, that aims at being as computationally inexpensive as possible [7]. The other is implemented in an under-development 3D Visual Analytics utility 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 user. 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. The results were of very modest effectiveness in the sense that it does not enable the user to grasp network features with ease, specially if the user is not musically trained. Nevertheless, it made the (audio)visualization more interesting, 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 believe valid a description of the sonification as it was left implemented, i.e. in its final version: the lower noise has central frequency proportional to first hub connectivity, at the extreme left on the layout, on the start of the sine, and it updated each two tempos. The higher noise has central frequency proportional to the second hub, and is also updated every two tempos. The third hub connectivity is proportional to the lower note, which is incident once a beat. The fourth hub connectivity is proportional to the higher note, one note occurring twice every half-beat.



(a) Versinus in JavaScript A (b) Versinus in JavaScript B

Fig. 3: The two implementations of Versinus in JavaScript using WebGL. In (a) is an image of the implementation in ccNetViz, a library with the purpose of being computationally inexpensive. This implementation is only for static networks. In (b) is an image of the Versinus implementation in a Visual Analytics utility being developed. This latter software render the animations in real time, with multiple controls for inspecting the network struture and controlling the animation, in conjunction with other analysis techniques. Also, the sine oscilates both in the vertical and depth axes, as does the periphery segment. Further information is given in Section 3.

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 flavours of Versinus. Some of these animations are available in six playlists of Youtube videos. For organization and brevity, they are available at [2] together with other Versinus-related resources.

4 Insights gathered

Valueable insights were obtained by the design and use of Versinus. For example, the visualization of network structures are favoured by the positoning of nodes in an oscillatory pattern. 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 [8]. The use of only one period for the sine is arbitrary and was established because it entailed good visualizations for the networks and tasks at hand during the desing of Versinus, but is allowed to vary in more recent implementations. Another important insight is the pertinece of fixed positions for the observation of evolving networks. Not only this has the potential to help ammeliorate the hairball effect [12], 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, as in a force-based layout. The positioning 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 dependant behaviors. Visual cues such as node width, height, shape and color were found informative for individual nodes but resulted in a cognitive overload for the observation of global features, which motivates simplified versions. The sonification of the evolving network added to the exotic flavour of Versinus and made it more attractive. It seems that more informative sonifications will require a dedicated work, 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) are very dependent of the network and intended tasks, and calls for interactive interfaces and automated indication of initial values.

5 Conclusions and further work

Within this document, we achieved a first thorough description of the Versinus method, with its formal elements, use context and software implementations. The validity of Versinus is implied both by its use in published research scenarios and the insights it yielded on individual networks and network layouts in general. Further developments using the method are already being made, with focus on interactivity and Web technologies, such as WebGL and the use of well-polished JavaScript libraries.

Next steps include automated tunning 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 [8]. Most importantly, we aim at completing the Visual Analytics utility, enabling the use of Versinus in an interactive environment, for enhanced exploratory analysis. Finally, we plan to further analyze datasets of evolving networks [4] 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. Fabbri, R.: The git repository for indicating versinus resources (2019), https://github.com/ttm/versinus
3. 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)
4. 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)
5. Fenu, C., Higham, D.J.: Block matrix formulations for evolving networks. SIAM Journal on Matrix Analysis and Applications **38**(2), 343–360 (2017)
6. Fisher, D.: Animation for visualization: opportunities and drawbacks. Ch **19**, 329–352 (2010)

7. 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)
8. 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)
9. 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)
10. Munzner, T.: Visualization analysis and design. AK Peters/CRC Press (2014)
11. 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)
12. Nocaj, A., Ortmann, M., Brandes, U.: Untangling hairballs. In: International Symposium on Graph Drawing. pp. 101–112. Springer (2014)
13. Palla, G., Barabási, A.L., Vicsek, T.: Quantifying social group evolution. Nature **446**(7136), 664 (2007)
14. Peel, L., Clauset, A.: Detecting change points in the large-scale structure of evolving networks. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
15. 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)
16. Ware, C.: Information visualization: perception for design. Elsevier (2012)
17. 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)